MASTER EN SCIENCES INFORMATIQUES

Élaboration d'un logiciel auteur destiné au domaine du handicap

Vandenabeele, Luc

Award date:
1993

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique
rue Grandgagnage, 21, 5000 Namur

Elaboration d'un logiciel auteur destiné au domaine du handicap

Luc VANDENABEELE

Promoteur: Madame le Professeur Monique NOIRHOMME-FRAITURE

Mémoire présenté en vue de l'obtention du grade de
Licencié et Maître en Informatique

Année académique 1992-1993

Résumé

Ce travail est la continuation d'un projet qui a démarré l'an passé par le mémoire de Stéphanie Baudrenghien et de Rudy Démo intitulé "Elaboration d'une boîte à outils d'aide à la réalisation d'interfaces pour personnes handicapées"[DEMO,92].

L'objectif de ce travail est l'élaboration d'un logiciel auteur permettant la réalisation de petits exercices -mélangeant images, sons et animations- destinés à des enfants handicapés. Toute l'originalité de ce projet repose sur le fait que ces exercices ne devront plus être implémentés par un informaticien mais directement par les éducateurs encadrant ces enfants. Le caractère graphique de l'interface du logiciel développé, ***Auteur!****, rend son utilisation très intuitive et diminue considérablement le temps nécessaire à l'implémentation des exercices.*

*Pour atteindre cet objectif, nous avons eu recours à des méthodes d'évaluation de logiciel afin de tester la boîte à outils. La conception d'****Auteur!*** *a été réalisée selon une méthode orientée objet et son implémentation en* ***C++*** *a nécessité l'apprentissage des techniques de programmation* ***Windows****.*

Abstract

This work is the continuation of a project begun last year with the thesis of Stéphanie Baudrenghien and Rudy Démo headed "Elaboration d'une boîte à outils d'aide à la réalisation d'interfaces pour personnes handicapées"[DEMO,92].

The aim of this work consists in the elaboration of an authoring software which enables to create small exercises including pictures, sounds and animations meant for disabled children. The fact that these exercises can be directly implemented by the educators taking care of the children instead of a computer scientist makes the very uniqueness of this project. The graphic interface of the software ***Auteur!*** *makes it very simple to use and considerably speeds up the implementation of the exercises.*

This objective has been reached with the help of software assesment methods which enabled us to test the tool box. ***Auteur!*** *was developed according to object oriented methods and the implementation in* ***C++*** *required to learn* ***Windows*** *programming techniques.*

Remerciements

C'est pour moi un plaisir d'exprimer ma plus vive reconnaissance à toutes les personnes qui ont collaboré, de près ou de loin, à la réalisation de ce mémoire.

Je tiens à remercier plus particulièrement:

Madame le Professeur Monique Noirhomme-Fraiture, promoteur de ce mémoire, pour l'aide, les encouragements et les conseils précieux qu'elle m'a prodigués tout au long de l'élaboration de ce travail.

Mesdames Quintin et Vandiependael, les logopèdes du Trèfle, ainsi que Madame G. Fraiture, pour leur accueil chaleureux durant le stage à Chièvres et la visite à la Croix Blanche de Bastogne.

Rudy Démo et Stéphanie Baudrenghien, les étudiants qui ont commencé le projet, pour leurs explications techniques concernant la boîte à outils développée l'an passé.

Je tiens également à remercier mes parents qui m'ont permis d'entreprendre ces années d'études et d'aboutir ainsi à l'élaboration de ce mémoire.

Merci.

Table des matières

Introduction

L'Institut d'Informatique des Facultés Notre-Dame de la Paix reçoit, depuis quelques années déjà, des cahiers des charges provenant d'institutions pour personnes handicapées.

A ce jour, plusieurs applications telles que ***OrdiThéâtre***[1]***,*** ***Comptes***[2]*, ou* ***Corps humain***[3] *ont été conçues et développées, sous la direction de Madame Noirhomme, par des étudiants en Informatique.*

L'objectif de ce travail est de permettre la création, par les éducateurs eux-mêmes, de petits exercices destinés aux enfants handicapés.

La première phase du projet a débuté l'an passé avec le mémoire de Stéphanie Baudrenghien et de Rudy Démo[4]*. Elle consistait en l'élaboration d'outils informatiques généraux formant les éléments d'une interface spécifique au handicap.*

La phase actuelle du travail consiste à intégrer ces outils dans un logiciel facilitant leur manipulation afin de créer interactivement les exercices destinés aux enfants.

---

[1] *cfr [BROU,90].*
[2] *cfr [DEPL,89].*
[3] *cfr [LEP,90].*
[4] *cfr [DEMO,92].*

Dans le premier chapitre, nous proposerons un rappel des différents outils développés l'an passé par Stéphanie et Rudy et nous donnerons un aperçu des exercices conçus à partir de la boîte à outils.

Durant le stage au Trèfle et à La Croix Blanche, les outils ont fait l'objet d'une évaluation qui a permis de retirer plusieurs remarques. Le deuxième chapitre présentera cette évaluation.

L'objectif de ce travail est de mettre directement les outils à la disposition des éducateurs encadrant les enfants handicapés. Pour atteindre cet objectif, il a fallu élaborer ce qu'on appelle un logiciel auteur. *Le chapitre trois présentera cet aspect de* logiciel auteur *en soulignant entre autre les différentes catégories et les domaines d'application de tels logiciels. La fin de ce chapitre sera consacrée aux caractéristiques que devrait posséder un logiciel auteur destiné au domaine du handicap.*

Plusieurs éléments sont venus agrandir la boîte à outils. Afin de garder une certaine homogénéïté par rapport au travail effectué l'an passé, l'analyse orientée objet de ces nouveaux outils a été modélisée grâce au langage OBLOG. Cette analyse fera l'objet du quatrième chapitre.

Les deux chapitres suivants se rapporteront au choix de l'environnement de développement et aux problèmes rencontrés durant l'implémentation du logiciel: maintenance des outils, architecture des classes C++, techniques de programmation Windows, etc.

Le chapitre sept montrera, par un exemple, les étapes à suivre pour créer un exercice avec le logiciel auteur.

Le dernier chapitre de ce travail présentera les perspectives du projet.

Chapitre 1

Les éléments de la "boîte à outils"

Le projet initial de ce travail a débuté l'année passée avec le mémoire de Stéphanie Baudrenghien et de Rudy Démo intitulé ***"Elaboration d'une boîte à outils d'aide à la réalisation d'interfaces pour personnes handicapées"****. Ces deux étudiants ont, dans un premier temps, analysé les mémoires existant dans le domaine du handicap afin d'en retirer les traits communs. Ils ont ensuite déterminé une série d'outils à offrir à un informaticien et les ont implémentés.*

Certains exercices provenant du cahier des charges du Trèfle ont été réalisés grâce à cette boîte à outils. Les quelques pages qui suivent présentent brièvement les outils ainsi que plusieurs exercices testés.

1.Présentation des outils

Les premiers outils qui ont été retenus lors de l'analyse de Rudy et de Stéphanie sont les suivants:

- la liste fixe,
- le dérouleur,
- la boîte de dialogue,
- la boîte d'édition,
- l'écran,
- l'icône.

Ces outils ont été conçus selon une méthodologie ***orientée objet*** (O.O.). L'utilisation d'une telle méthodologie semble adéquate et naturelle dans le contexte qui nous occupe. En effet, nous pouvons considérer ces outils comme des ***objets*** graphiques auxquels sont attachées des ***opérations***. L'idée de centrer la structuration du programme autour des données en leur associant les traitements qui leur sont spécifiques en découle directement. Notons encore que les outils ont été élaborés sous l'environnement Windows qui encourage la programmation O.O. .

A la fin de la première phase du projet[1], les outils étaient regroupés sous forme de librairie. Ils étaient destinés à un informaticien qui pouvait les utiliser dans un programme C.

Revenons brièvement sur chaque objet de la boîte à outils. Le lecteur intéressé pourra consulter [DEMO,92] pour de plus amples informations.

1.1.La liste fixe

La liste fixe permet d'afficher à l'écran une liste d'éléments. Ces éléments seront soit des libellés, soit des pictogrammes. Aucun défilement n'est possible. Plusieurs paramètres doivent être déterminés par le concepteur de l'application. A savoir : la localisation, la couleur de fond, le nombre d'éléments ainsi que les différents éléments de la liste.

[1] *objectif du mémoire de Rudy et de Stéphanie*

tomates
oignons
poireaux
céleri

Figure 1: *Exemples de listes fixes*

1.2.Le dérouleur

Lors d'un dialogue, l'utilisateur est souvent amené à faire un choix entre plusieurs éléments. Il arrive que, faute de place sur l'écran, le concepteur ne puisse présenter les différentes propositions en une seule fois. Une solution largement rencontrée dans les applications pour personnes dites normales est ce que l'on appelle la liste de sélection (cfr. figure 2) . Cependant, un tel objet semble trop compliqué pour la population envisagée. C'est pourquoi il a été décidé de proposer une version simplifiée et mieux adaptée de cet objet: le ***dérouleur.*** C'est une liste déroulable d'éléments parmi lesquels l'utilisateur a la possibilité de faire une sélection. Les éléments, tout comme pour la liste fixe, peuvent être des libellés ou des pictogrammes. La sélection se fait soit en cliquant sur les boutons "haut" et "bas" soit de manière directe, c'est-à-dire sur l'élément choisi. Un dérouleur possédera toujours un élément "vide" contrasté afin d'éviter toute perturbation de la personne handicapée par un élément contrasté d'avance. Il va de soi que cet élément est insélectionnable.

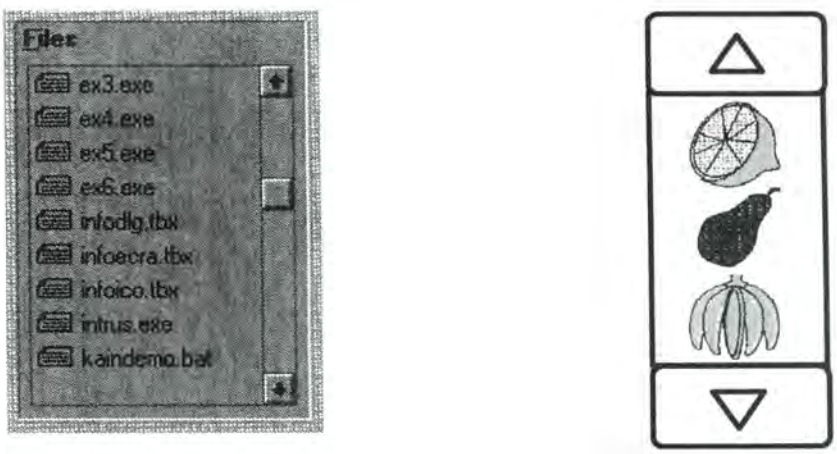


Figure 2: *A gauche, une liste de sélection classique. A droite, le dérouleur.*

Pour décrire un dérouleur lors de la conception d'une application, on pourra déterminer la taille des flèches de défilement, leur couleur de fond, la couleur de leur libellé, le type des éléments du dérouleur, la couleur de fond de ces derniers. Il faudra également donner la localisation du dérouleur, sa taille et le nombre total de ses éléments. De nouveau, toutes ces caractéristiques devraient permettre une adaptation à la population concernée.

1.3.La boîte de dialogue

La boîte de dialogue permet un retour d'information à l'utilisateur. Ce retour d'information est d'une utilité capitale pour l'utilisateur puisqu'il lui permet dans de nombreux cas d'éviter des erreurs ou de les corriger. Pour un utilisateur qui présente une déficience mentale, cette utilité est encore accrue. En effet, de tels utilisateurs doivent être fortement guidés et le dialogue sera essentiellement dirigé par les besoins de l'application. L'utilisateur sera, quant à lui, essentiellement réactif. La boîte de dialogue implémentée ici sera dès lors de type modale. C'est-à-dire que l'utilisateur ne pourra continuer sa tâche tant qu'il n'aura pas agi dans la boîte. De manière générale, la boîte de dialogue permettra à l'utilisateur de réfléchir sur ses actions et sur les conséquences de celles-ci. Il faudra, cependant, faire attention au danger de distraire ou de démotiver l'utilisateur.

Figure 3: Dans ce cas-ci, la boîte de dialogue annonce que la réponse donnée par l'utilisateur est correcte .

Les boîtes de dialogue proposées sont composées d'un libellé, d'un pictogramme et de un ou deux boutons grâce auxquels l'utilisateur confirme la lecture du message ou répond à une question.

1.4.La boîte d'édition

Il est parfois demandé à l'utilisateur d'introduire de l'information au clavier. Un objet interactif permettant cela a donc été créé. Un bouton de validation est associé au champ de saisie (cfr. figure 4).

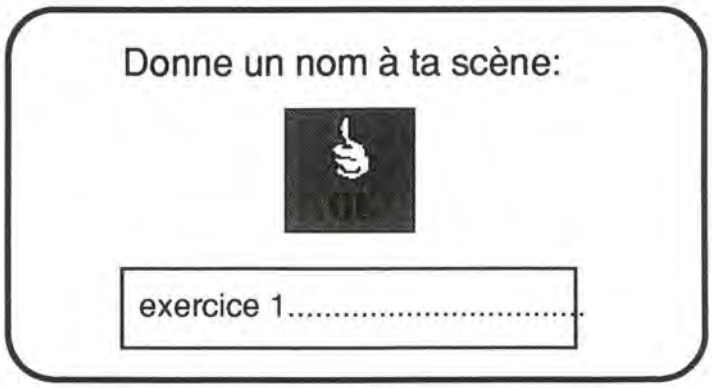


Figure 4: Une boîte d'édition invitant l'utilisateur à entrer le nom d'une scène.
(exemple repris du logiciel de Marionettes [BROU])

Bien que ce genre d'interaction est généralisé dans beaucoup de mémoires réalisés à l'Institut, il faut mettre en garde le concepteur sur les difficultés que cela peut engendrer pour une personne fortement handicapée. Une alternative proposée est ce qu'on appelle "l'écritoire" qui affiche à l'écran les lettres de l'alphabet. L'utilisation du clavier ne serait donc pas obligatoire.

1.5.L'écran

La conception des différents écrans d'une application doivent répondre à un minimum de principes: regroupements logiques des informations, homogénéité de représentation...
Dans le domaine du handicap, on peut observer que les écrans des applications sont souvent divisés en deux zones (cfr figure 5).

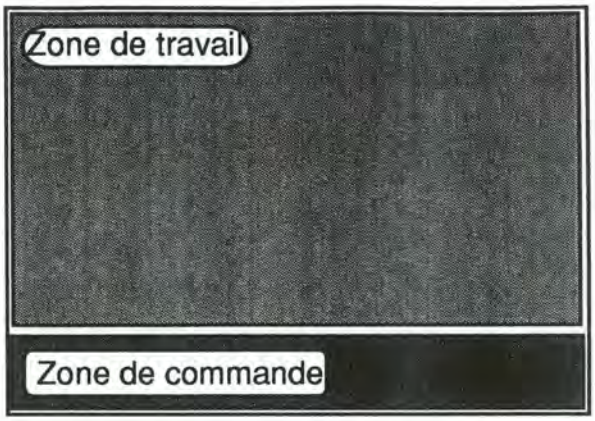


Figure 5: Un écran est généralement divisé en deux zones.

Nous avons, d'une part, la zone de travail qui reprend les informations relatives à la tâche elle-même. D'autre part, la zone de commande regroupe les informations concernant le déroulement de la tâche. Elle est constituée de boutons associés à des commandes dirigeant l'exécution de l'application.

1.6.L'icône

Le dernier outil présenté dans cette section n'en est pas le moindre. En effet, il reviendra dans presque tous les exercices développés avec la boîte à outils. Il s'agit de l'icône.

Figure 6: Quelques pictogramme repris des exercices du cahier des charges du Trèfle.

De nouveau, cet objet peut être paramétré par le concepteur de l'application. L'icône a une représentation graphique. C'est celle d'un pictogramme (cfr. figure 6).
Une icône possède également des propriétés:

- ***(In)sélectionnable***: l'objet peut ou non être sélectionné par la personne handicapée,
- ***(In)amovible***: l'objet peut ou non être déplacé durant l'exécution du programme,
- ***(In)visible***: l'objet peut être visible ou non pendant le traitement,

- ***(Non)pile***: l'objet peut être dupliqué ou non par une simple sélection lors de l'exécution du programme,
- ***Défaut***: cette propriété signifie que l'icône est sélectionnable, inamovible et visible.

2.Description de quelques exercices testés

Grâce aux outils décrits précédemment, on peut concevoir une multitude de petits exercices. Cette section a pour but d'en présenter plusieurs afin d'éclairer le lecteur sur les possibilités de la boîte à outils réalisée l'an passé.

2.1.Apprentissage des couleurs

Le premier exercice concerne l'apprentissage des couleurs. L'écran présente un fruit colorié qui sert de modèle, un fruit à moitié colorié ainsi que plusieurs couleurs. Il est demandé à l'utilisateur de choisir la couleur correcte afin de compléter le fruit à moitié colorié. Tant que l'utilisateur n'a pas choisi la bonne couleur, une boîte de dialogue l'invite à recommencer son choix.

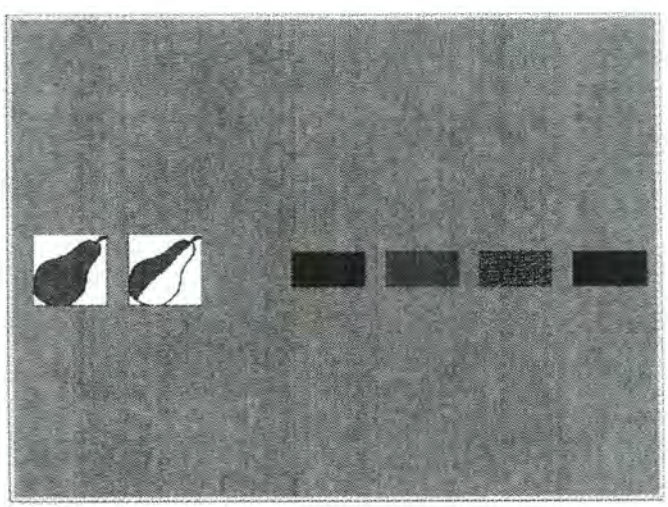


Figure 7: Interface de l'exercice d'apprentissage des couleurs.

Plusieurs versions de cet exercice ont été implémentées. Chacune d'elles varie par le nombre de couleurs proposées, par la présence ou pas d'un modèle, etc.
Pour concevoir cet exercice, il a fallu utiliser principalement l'objet icône avec différentes propriétés:
Pour les modèles des fruits, les icônes étaient insélectionnables alors que pour les couleurs, celles-ci devaient être quant à elles sélectionnables.

2.2.Intrus

Le deuxième type d'exercice consiste à retrouver parmi plusieurs pictogrammes celui qui n'a pas de lien avec les autres. Dans cet exemple, un pictogramme représentant un poulet

a été introduit au milieu de pictogrammes figurant des fruits. L'utilisateur est invité à faire son choix en cliquant sur un des pictogrammes. Une boîte de dialogue apparaît alors à l'écran indiquant le résultat de la sélection. Dans le cas d'une bonne réponse, l'exercice se termine automatiquement.

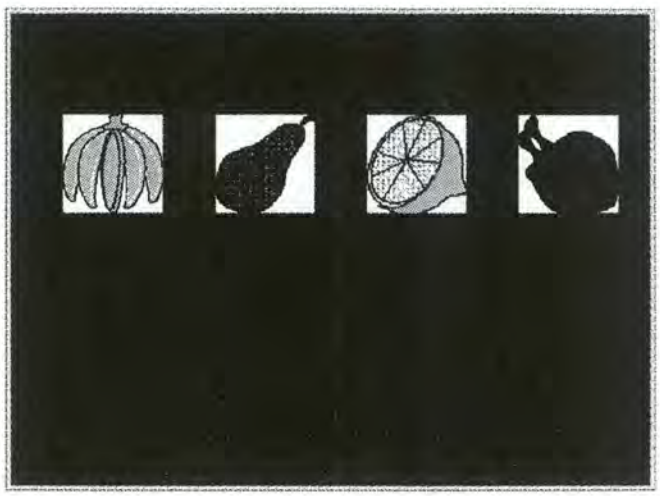


Figure 8: Interface de l'exercice de recherche d'intrus.

La modélisation de l'exercice avec les objets de notre boîte à outils est assez aisée puisque il suffit de représenter chaque pictogramme par une icône ayant la propriété sélectionnable.

2.3.Elaboration de suites

L'exercice proposé ici consiste à réaliser une suite logique. L'utilisateur doit sélectionner et positionner certains objets graphiques pour former la suite d'objets. L'exercice se présente comme indiqué sur la figure 9.

Figure 9: Interface de l'exercice d'élaboration d'une suite.

Lorsque l'utilisateur dépose son objet près de la suite, une zone d'imprécision de 64 pixels sur 126 est acceptée. Ceci afin de ne pas décourager l'utilisateur ayant des difficultés lors de la manipulation de la souris. Si l'objet est placé dans cette zone, il sera ajusté correctement à la fin de la suite. En outre, un objet ne complétant pas correctement la suite, même s'il est placé dans cette zone, disparaîtra de l'écran afin de ne pas l'encombrer inutilement.
L'exercice a été construit de la manière suivante:

- La partie de la suite qui sert d'exemple est représentée par une icône insélectionnable.
- Les deux éléments disponibles pour compléter la suite sont des icônes ayant la propriété pile.
- Enfin, une icône de validation permet de terminer l'exercice. Cet objet est, bien entendu, sélectionnable.

Le lecteur intéressé pourra visualiser d'autres scénarios d'exercices dans [DEMO,92].

Chapitre 2

Evaluation de la "boîte à outils"

Ce chapitre reflète d'une part les réflexions et les remarques des logopèdes du Trèfle qui avaient proposé un feed-back des exercices (évaluation avec des enfants), et résume d'autre part l'entrevue avec Madame G. FRAITURE, logopède à la Croix Blanche située à Bastogne.

1.Quelques mots sur l'évaluation de logiciels pour personnes handicapées

Dans "***Evaluation of software for people with mental disabilities***" [NOIR,91], Madame Noirhomme, qui supervise depuis plusieurs années des mémoires dans le domaine du handicap, traite des questions telles que :

- Pourquoi faire une évaluation ?
- Sur quoi doit porter cette évaluation ?
- Comment la réaliser ?

Regardons maintenant plus en détails les cinq points principaux qui permettent une bonne évaluation:

1.1.L'objet de l'évaluation

Le premier point sur lequel il faut attirer l'attention est l'objet de l'évaluation. Faut-il évaluer le **logiciel** en lui-même (sa fiabilité, sa convivialité...) ou l'**utilisateur** grâce au logiciel (ses connaissances, ses réactions affectives et psychologiques ...)?

Une telle évaluation de l'utilisateur est un travail pour un psychologue. Dans notre cas, nous nous attarderons plutôt sur le travail du concepteur du logiciel qui désire livrer un produit qui rencontre la demande.

1.2.Les objectifs du logiciel

Une fois que l'objet de l'évaluation est défini, il est nécessaire de spécifier les objectifs du logiciel:

Est-ce un outil aidant l'utilisateur dans sa vie quotidienne, un logiciel didacticiel, un outil pour les psychologues ou plus simplement un moyen de familiariser l'utilisateur à l'ordinateur ?

Tous ces objectifs se sont pas contradictoires mais il est bon de les classer selon les priorités des exigences du logiciel.

En plus de ces objectifs, viennent s'ajouter les utilisations "déviées" du programme. Celles-ci sont presque inévitables et ne sont pas seulement présentes dans le domaine informatique.

1.3.Le profil de l'utilisateur

Une dernière chose à prendre en compte est le profil de l'utilisateur du logiciel. Cela englobe son habileté, ses aptitudes, sa façon de penser, son vocabulaire, etc...

L'évaluation vérifiera si le logiciel rencontre ses objectifs et s'il est bien adapté à la population concernée.

1.4.Les différents niveaux de l'évaluation

On peut entreprendre l'évaluation à trois niveaux:

- La **fiabilité** du logiciel

Cette partie de l'évaluation concerne ce qu'on appelle en informatique la correction d'un programme. Le concepteur vérifie que son programme est fiable. Pour cela, il existe des méthodes de preuve de programme. On peut également, dans la phase qui suit l'implémentation, effectuer une série de tests qui couvrent toutes les situations possibles. Ceux-ci devraient donc détecter les erreurs. Malheureusement, on ne peut pas vraiment être sûr de la complétude de ces tests... Cette remarque reste générale en informatique et ne concerne donc pas seulement les logiciels du domaine du handicap.

- La **convivialité** du logiciel

La convivialité du logiciel est le maître-mot de l'évaluation. Cette caractéristique que les concepteurs de logiciels tendent de plus en plus à atteindre est particulièrement importante dans le domaine du handicap. Les méthodes d'évaluation concernant la convivialié font l'objet du point suivant.

- Les **impacts** sur l'utilisateur

L'utilisation de tels programmes a une influence sur les personnes présentant des déficiences. Il est dès lors important de pouvoir mesurer également les impacts des logiciels sur les utilisateurs. Le logiciel concerné joue-t-il bien son rôle (apprentissage, aide, ...) ?

Bien sûr, des effets inattendus peuvent survenir. Il est intéressant de pouvoir les identifier car ils peuvent provoquer, dans certains cas, des troubles chez les utilisateurs.
Malheureusement, ce genre d'évaluation demande énormément de temps et est, dans bien des cas, délaissée.

1.5.Les outils d'évaluation

A l'Institut d'Informatique, plusieurs méthodes sont utilisées pour évaluer un programme:

- Les **questionnaires**:

Le questionnaire, que l'on appelle également "grille d'évaluation" - est composé de plusieurs questions que la personne menant l'expérience est tenue de compléter. Une grille standard est divisée en trois parties et teste la convivialité du logiciel:

√ Première partie:

Cette première partie concerne les capacités de l'utilisateur: son identification, ses capacités mentales ainsi que ses aptitudes dans le domaine relatif au logiciel. Ces questions sont à remplir avant l'utilisation du logiciel par des personnes qui connaissent bien l'utilisateur.

√ Deuxième partie:

Cette partie identifie les personnes qui dirigent le test. Cela permet de relativiser certains résultats...

√ Troisième partie:

Les questions que l'on trouve dans cette partie sont à remplir durant les séances d'observations et concernent:

- ***le logiciel en général:*** cela reprend des questions qui sont valables pour l'ensemble du logiciel et que l'on pose une fois pour toutes;
- ***les écrans*** : ce sont les questions à propos des différents écrans de l'application;
- ***l'attitude de l'utilisateur:*** il est important d'observer le comportement de l'utilisateur durant une session. Il se peut qu'il soit troublé par quelque chose n'ayant rien à voir avec le logiciel, ce qui fausse bien entendu le test;
- ***l'opinion de l'utilisateur***: il est intéressant de poser des questions directement à l'utilisateur - dans la mesure du possible - afin de ressentir ce qu'il pense du logiciel;
- ***les remarques des personnes dirigeant le test:*** les questions posées ici concernent la convivialité, l'utilité du logiciel etc...

L'utilisation d'un tel questionnaire permet très rapidement de remanier le logiciel en y apportant quelques modifications. Afin d'obtenir des résultats facilement comparables, il convient de bien choisir l'échantillon des utilisateurs observés, de noter la fréquence des

tests effectués ainsi que l'identité des expérimentateurs. Le niveau d'aide prodigué à l'utilisateur durant les différentes sessions doit également être noté.

- Les **traces** :

Toutes les actions effectuées lors d'une séance peuvent être enregistrées formant ainsi ce qu'on appelle la trace de l'exécution du programme. Telles quelles, les traces n'apportent rien pour l'évaluation d'un programme. Ce n'est qu'une fois interprétées par un autre programme que l'on peut en retirer quelque chose. On peut alors déceler, par exemple, un taux d'erreur anormal ou encore l'existence de difficultés dans certaines parties du logiciel. Un facteur important que l'on retrouve dans les traces est le temps écoulé entre deux commandes. Toutes ces observations peuvent amener le concepteur du logiciel à revoir telle ou telle partie de son programme.

Bien sûr, les traces se sont fiables que lorsque l'utilisateur ne reçoit pas d'aide de la part de la personne qui l'encadre.

- Les **enregistrements** audio-visuels :

Cette méthode permet d'avoir un avis différent de l'initiateur de la session et par cela d'avoir un regard plus neutre. Les séances d'exercices sont filmées et analysées par la suite.

1.6.Le projet MUSiC

L'évaluation de logiciel est un domaine prenant de plus en plus d'ampleur. Et cela pour plusieurs raisons. Elle donne, en premier lieu, une garantie aux utilisateurs potentiels qui peuvent être assurés de la qualité des logiciels vérifiant les critères de l'évaluation. Ensuite, si on la considère comme partie intégrante du cycle de développement de logiciel, l'évaluation permet d'apporter des modifications, voire même des améliorations, à un moment où celles-ci peuvent se faire assez facilement et sans trop de frais.

Ce désir de qualité du produit est supporté par des directives européennes et des standards internationaux. Ainsi, le projet 5429 d'ESPRIT II, 'MUSiC' (l'abréviation de ***Metrics for Usability Standards in Computing***), s'occupe de développer des métriques, des méthodes et des standards requis pour l'évaluation de logiciels. Huit partenaires provenant de six pays européens sont impliqués dans ce projet dont les deux objectifs principaux sont, d'une part, le développement de métriques utilisées dans l'évaluation de logiciels et, d'autre part, la dissémination des concepts et des méthodes de MUSiC dans l'industrie.

Le projet englobe quatre types de mesures applicables à différents moments du cycle de développement:

- ***Les mesures analytiques***: Elles sont applicables très tôt dans le cycle de la conception puisqu'elles demandent au minimum des spécifications du produit sur papier. L'objectif est de tester les spécifications le plus tôt possible afin de réduire au maximum les modifications du projet. Ces mesures analytiques ne peuvent se faire que dans des domaines où des représentations ***formelles*** du système existent.

- ***Les mesures de performances***: Elles sont utilisées lorsque une simulation ou un prototype du système est disponible. Les utilisateurs sont observés lorsqu'ils exécutent des tâches représentant le travail pour lequel le système est prévu. Les performances sont mesurées par des observations objectives[1]. Ces tests utilisés de manière itérative durant le cycle de développement permettent de rafiner la conception du produit final.

- ***Les mesures du travail cognitif***: Ces mesures sont également appliquées itérativement durant le cycle de développement dès qu'un prototype du système est disponible. Les mesures prises concernent l'effort demandé aux utilisateurs lorsque ceux-ci doivent manipuler le système. Le ratio de performance à l'effort est alors utilisé comme un indicateur de la qualité du logiciel.

- ***Les mesures de la satisfaction de l'utilisateur***: Elles sont applicables lorsqu'il existe une version exécutable du logiciel[2]. Les utilisateurs testent le système et remplissent un questionnaire reflétant leur avis sur le logiciel.

Après avoir été développées et testées en laboratoires, ces quatre types de mesures seront utilisés dans le domaine industriel et des méthodes pour les interpréter seront alors disponibles.

Terminons cette section par un exemple d'outil facilitant l'évaluation assistée par vidéo: DRUM (Diagnostic Recorder for Usability Measurement).

---

[1] *Autrement dit: des observations basées sur des règles.*

[2] *Version précédente, prototypes, version Béta...*

Ce logiciel - qui tourne sur l'Apple Macintosh et exige un magnétoscope compatible - réduit considérablement le temps consacré à l'analyse d'une vidéo.

Il permet:

- Le contrôle des données durant toute la période d'évaluation,
- Le contrôle de la vidéo et la création d'un fichier enregistrant les événements se produisant durant chaque session d'évaluation,
- La recherche et la vision automatique de n'importe quel événement enregistré,
- L'analyse de données enregistrées.

L'écran général donne une vue d'ensemble des fonctionnalités de DRUM (cfr figure 10). L'utilisateur a un accès direct sur chaque module du système et peut obtenir à tout moment une aide "on-line".

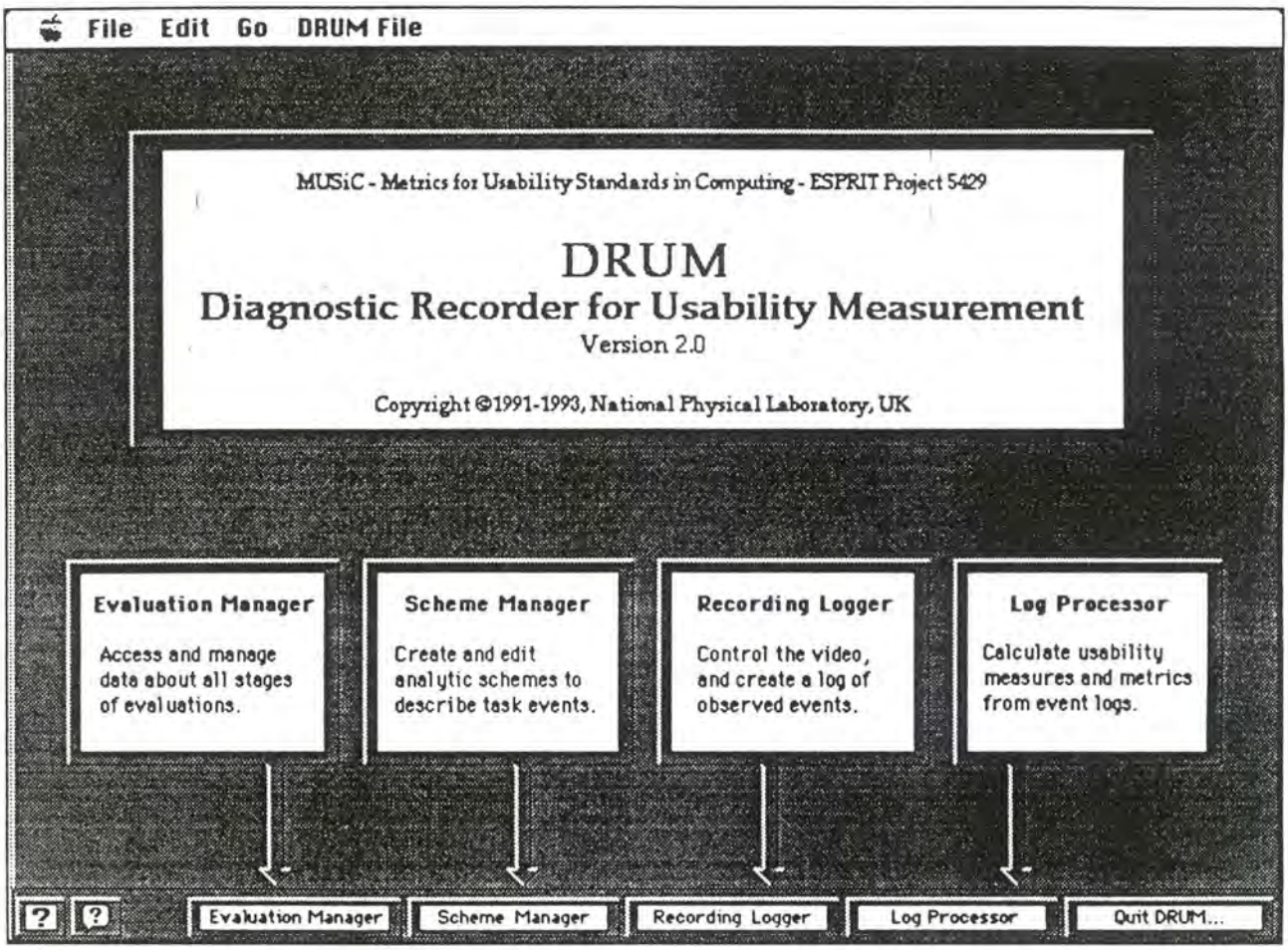


Figure 10: Ecran principal de DRUM

source:Software support for video-assisted evaluation of usability.DRUM, Miles Macleod NPL,April 93

2.Evaluation des outils

Durant le stage, qui s'est déroulé du mois d'octobre à décembre, une première évaluation des outils décrits en début de chapitre a été entreprise. Cette évaluation n'a certe pas la prétention d'être complète: il aurait fallu pour cela plus de temps et l'étendre à une population plus importante. Toutefois, elle a fait ressortir quelques remarques intéressantes.

2.1.Le Trèfle

- **Présentation**

Le Trèfle est situé à Kain (dans la région tournaisienne). Une antenne a été construite à Chièvres. Ce centre accueille des enfants qui ont des problèmes de psycho-motricité ou qui ont des déficiences mentales. Depuis peu, il existe également une section pour de jeunes autistes...

Les logopèdes du Trèfle, Mesdames Quintin et Vandiependael, avaient proposé un ensemble d'exercices simples d'acquisition de connaissance. Certains d'entre eux ont été réalisé[3] et les notes qui suivent reflètent les différentes expérimentations avec les enfants du centre.

- **Expérimentation et évaluation**

Nous avons pris rendez-vous tous les lundis matin (jusqu'en décembre) pour expérimenter les exercices avec quelques enfants. L'évaluation, qui s'est faite sur base d'un questionnaire, portait sur plusieurs points.

En premier lieu, bien sûr, il s'agissait de tester l'adéquation des différents outils présentés en début de chapitre: fiabilité, modifications à apporter, complétude etc.

Un autre point à évaluer concernait les moyens d'interface avec les différents exercices.

[3] *cfr. chapitre 1*

Au-delà de l'évaluation des outils proprement dits, il était également intéressant de prendre en compte les remarques concernant l'élaboration des exercices par les moniteurs eux-mêmes. Comment pourraient-ils "construire" ou modifier eux-mêmes les exercices et cela simplement et rapidement ?

Au terme de ces séances, il a été possible de noter les points suivants:

GRAPHISME:

- Un des problèmes rencontrés est sans conteste la qualité des images affichées. Il faudrait donner la possibilité d'importer facilement des images soit créées dans un autre logiciel, soit digitalisées par un scanner.

- Des modifications sur ces images devraient être permises: position sur l'écran, agrandissement, réduction, rotation... etc... Ceci permettant une meilleure adaptation à l'utilisateur.

- Un autre point à relever concerne l'efficacité (au point de vue informatique) des outils graphiques. Ce problème dont la source provient de Windows sera traité dans le chapitre consacré à l'implémentation.

RENFORÇATEURS:

- Les renforçateurs ne sont sûrement pas à négliger dans de telles applications. Il est important que ceux-ci soient particulièrement bien adaptés à l'enfant qui utilise le programme. (l'enfant qui ne sait pas lire se 'moquera' de voir le message "BRAVO ! TU AS BIEN REPONDU!" apparaître à l'écran ...)

- Dans la mesure du possible, il serait intéressant de pouvoir inclure des sons ou encore de petites animations.

- Afin de ne pas perdre la portée des renforçateurs, il faudrait en associer plusieurs à une action[4]. Ceux-ci seraient appelés de manière aléatoire et l'enfant aura moins vite l'impression de "déjà vu".

INTERFACE:

- Le moyen de 'communiquer' avec les différents exercices est la souris (déplacement et sélection d'objets). Vu la variété des handicaps rencontrés, ce seul moyen ne suffit pas. En effet, certains enfants éprouvent beaucoup de difficultés à l'utiliser[5] et s'énervent assez rapidement. En conséquence, il semble impératif d'inclure d'autres méthodes d'interface. Celles-ci permettraient alors d'adapter l'accès des exercices à chaque enfant.

- En plus de l'interface spécifique pour chaque utilisateur, certaines combinaisons de touches du clavier devraient permettre à l'éducateur de garder le contrôle sur le déroulement de l'exercice: l'interrompre, le continuer, le terminer "prématurément", le recommencer depuis le début.

- Les logopèdes du centre demandent également que l'écran puisse rester "gelé" lorsqu'elles expliquent l'exercice à l'enfant. Pendant cette période, aucune action de l'enfant (click souris, touche du clavier enfoncée etc) ne devrait être prise en compte. Un système de blocage et déblocage des moyens d'interaction devrait donc être mis en oeuvre.

2.2.La Croix Blanche

- **Présentation**

La Croix Blanche est située à Bastogne. Ce centre accueille des enfants dont le handicap est quelque peu différent de ce que l'on peut trouver à Chièvres: les enfants de la Croix

---

[4] *=le fait de sélectionner la bonne ou la mauvaise réponse...*

[5] *que ce soit pour la déplacer ou cliquer sur le bouton sans faire bouger le curseur*

Blanche sont plus jeunes et ne présentent pour la plupart qu'un très léger handicap psycho-moteur .

Madame G. Fraiture, la logopède du centre, a présenté lors de notre entrevue les logiciels sur lesquels elle travaille avec les enfants. Je ne ferai pas ici une description de ces logiciels. Je me contenterai de signaler quelques remarques soulevées par Madame Fraiture lors de notre entretien. (Le matériel utilisé est un IBM PS2 équipé d'une carte sonore permettant la reconnaissance de la voix.)

- **Constatations**

Positives:

- La chose la plus frappante est sans conteste la présence de renforçateurs "amusants" dans la majorité des logiciels présentés: (une otarie qui joue avec une balle, une sorte de feu d'artifice à l'écran, etc). Il semble qu'une petite animation apporte beaucoup d'effets.

- Beaucoup de petits logiciels "amusants" existent (mais ils sont trop statiques!).

Négatives:

- Les logiciels présentés sont fermés. On ne peut donc pas les modifier, ni les adapter à un utilisateur particulier. Dans le domaine du handicap dont l'une des caractéristiques importantes est justement la diversité, ceci présente un grand défaut.

- Un autre point regrettable est le fait que l'éducateur ne peut pas interrompre un exercice commencé. Si l'enfant éprouve des difficultés ou si ,pour une raison quelconque, il ne désire plus "travailler" sur l'exercice , l'éducateur est "obligé" de le continuer lui-même jusqu'au bout.

3.Conclusion

Je conclurai ce chapitre par trois remarques:

La première concerne l'évaluation des outils. En général, ceux-ci sont bien adaptés et ils permettent de créer énormément d'exercices. Bien sûr, quelques modifications doivent être apportées pour obtenir une plus grande souplesse. Je pense, par exemple, à l'objet icône qui devrait pouvoir être redimensionné.

Il serait également intéressant d'ajouter un outil dérivé de l'icône permettant de petites animations. Le son devrait également être introduit dans la boîte à outils. Il semble que ces outils seraient fort utiles dans l'élaboration de renforçateurs.

La deuxième remarque s'applique à l'interface et aux moyens d'interaction. La seule manipulation de la souris est un obstacle difficile à surmonter pour certains utilisateurs. D'autres interfaces seraient donc les bienvenues: trackball, joysticks, système de défilement à un ou plusieurs 'switches'...

Enfin, la troisième remarque concerne l'élaboration d'un logiciel qui permettrait de construire relativement facilement les exercices. La population de ces deux centres est fort diversifiée. Par ailleurs, pour un même utilisateur, les logopèdes sont amenées à modifier la difficulté des exercices au cours du temps. L'idée principale à retenir est sans aucun doute la souplesse dont ce logiciel devra faire preuve: souplesse dans l'élaboration des exercices ainsi que dans leurs modifications. Sous ce terme se cache un compromis entre la facilité d'utilisation et la puissance du logiciel .

Chapitre 3

Les logiciels auteurs

L'objectif final du projet est d'offrir un logiciel permettant l'élaboration - par les logopèdes eux-mêmes - de petits exercices adaptés aux enfants handicapés. Les exercices peuvent être composés de pictogrammes, d'animations et de sons. Ce genre de logiciels que l'on appelle "logiciels auteurs" commencent à faire leur apparition sur le marché. Ce chapitre fera un tour d'horizon des logiciels auteurs existants et tentera de faire ressortir les caractéristiques principales que l'on retrouve dans ce genre de programmes. Une présentation des caractéristiques que devraient posséder un logiciel auteur destiné au domaine du handicap terminera ce chapitre.

1.Introduction

Cette idée de construire des logiciels facilitant l'élaboration de programmes n'est pas nouvelle. En effet, du chemin a été parcouru depuis le début de l'ère informatique où la programmation était vraiment une affaire de spécialistes. L'Homme devait connaître un langage facilement "interprétable" par la Machine. Maintenant, grâce aux progrès en interface Homme-Machine et aux outils qui existent sur le marché, cette programmation[1] lui est rendue plus aisée.

Avec l'émergence du multimédia, - *qui, selon la définition d'IBM, est "l'intégration élégante, au sein d'une configuration informatique, d'une série de technologies électroniques capable de produire, d'assembler et de délivrer des informations combinant le texte, l'image et le son"* - une nouvelle catégorie de logiciels devenait nécessaire. Il fallait des logiciels permettant d'exploiter, d'organiser, de lier entre eux les différents types de données tels que le texte, l'image et le son et de rendre cet ensemble interactif.

Ces logiciels sont constitués de groupes de macro-commandes interfacées le plus visuellement possible afin de faciliter le travail de l'utilisateur.

L'objectif du projet est de donner un logiciel facilitant l'élaboration d'exercices dédiés à des enfants handicapés. Nous avons donc, dans un premier temps, consulté la littérature afin de voir si les logiciels proposés sur le marché pouvaient répondre à notre demande.

2.Les facteurs influençant le choix d'un logiciel

Il existe une multitude de produits-logiciels spécialisés dans l'écriture de telles productions. Ils ont tous pour objectif de faciliter l'utilisation de l'informatique. Toutefois, la contrepartie de cette facilité d'utilisation est sans aucun doute la perte de généralité des produits en question. Aussi, en fonction de la nature du projet à mettre en oeuvre, de la population censée l'utiliser, on choisira le logiciel le plus proche de ses préoccupations. Le but de cette section est d'exposer les facteurs influençant le choix du logiciel auteur.

---

[1] *Il s'agit bien de la phase d'implémentation d'un programme et non pas de la phase de conception qui fait l'objet d'autres recherches.*

2.1.La structure de la production multimédia à concevoir

La structure de la production multimédia à développer est l'une des caractéristiques dirigeant le choix du logiciel. Cette structure peut être **linéaire** ou **non-linéaire**.

Les productions **linéaires** sont divisées en 3 parties: une d'ouverture, une de contenu et une de fermeture. En général, l'utilisateur ne peut modifier la séquence des sujets présentés. Toutefois, il est possible d'ajouter des boutons (ou autres objets sur lesquels l'utilisateur peut cliquer) pour visionner des détails supplémentaires sur le sujet courant (cfr. figure 1).

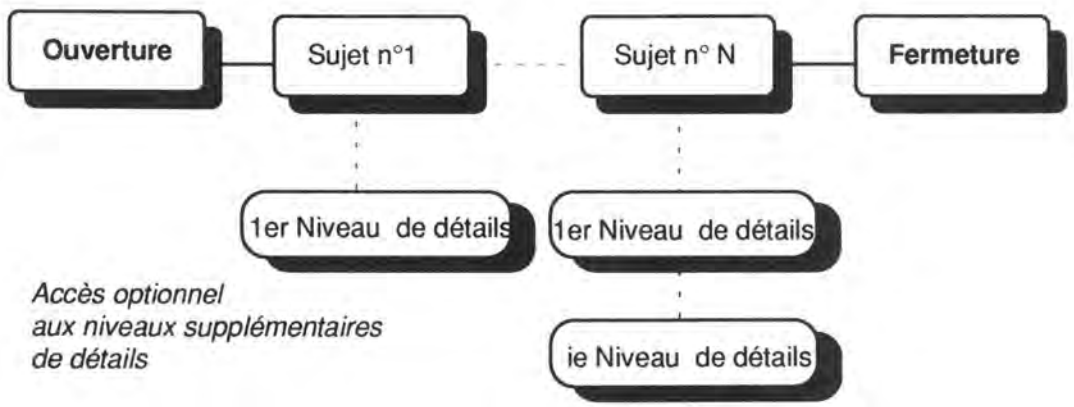


Figure 1: La structure d'une production multimédia linéaire.

Source: *Discover Windows 3.1.Multimedia, R.Jennings, 1992, Que Corporation ,Carmel,p553.*

La plupart des productions linéaires sont créées par des applications qui utilisent le principe de la ligne du temps.

Les productions **non-linéaires**, contrairement aux productions linéaires, donnent le contrôle de la séquence à l'utilisateur (cfr. figure 2). Les applications utilisées pour créer ces productions non-linéaires sont appelées ***logiciels auteurs de multimédia***[2]. Elles sont, comme on le dit en Anglais, *event-driven*. C'est-à-dire que des événements tels qu'un click de souris ou une touche de clavier enfoncée influencent le déroulement de la production.

[2] *En Anglais: Multimedia authoring software.*

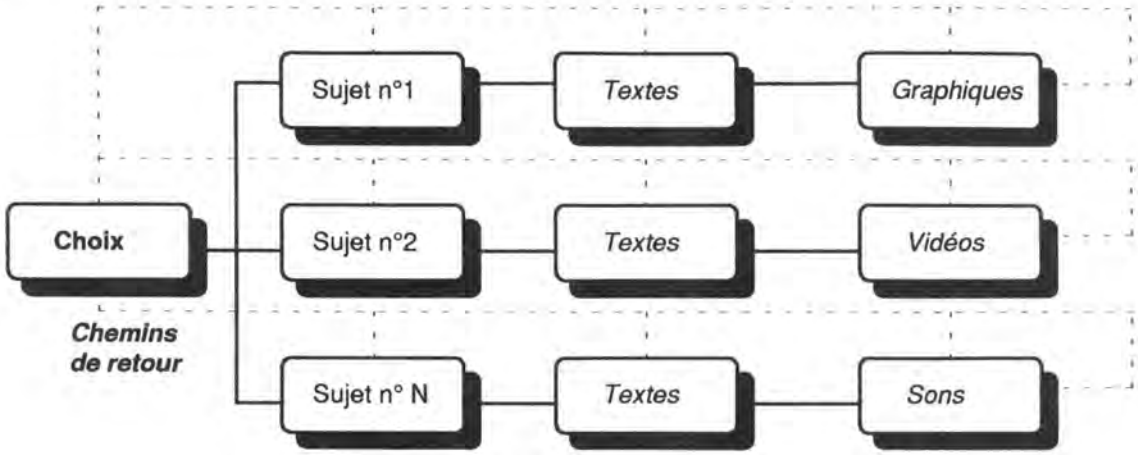


Figure 2: La structure d'une production multimédia non linéaire.

Source: *Discover Windows 3.1.Multimedia, R.Jennings, 1992, Que Corporation ,Carmel,p554.*

Un large éventail de logiciels auteurs existe aujourd'hui sur le marché. La plupart de ces applications simplifient la création des productions multimédias en améliorant l'interface homme-machine et en offrant des outils performants permettant l'intégration de textes, de graphiques, de sons, voire même de vidéo.

On peut classer ces logiciels en trois grandes familles selon le degré de facilité d'utilisation:

- **La programmation traditionnelle assistée par une interface graphique** (*mode programmation*) : Dans ce type de logiciels, les différents éléments constitutifs de l'interface peuvent être placés sur l'écran grâce à un éditeur. Après cela, il faudra encore programmer toutes les fonctions décrivant le comportement des objets interactifs.

- **La programmation sous forme de liste** (*mode script*) : Ici, chaque objet interactif est directement lié à un *script* qui décrit son comportement. Comme pour le premier type, chaque objet de l'interface peut être placé de manière directe sur l'écran.

- **La programmation du flux par organigramme** (*mode icônes*) : Avec ce troisième type de logiciels, bâtir une application est aussi simple que dessiner un organigramme. Chaque fonction est représentée par une icône et est paramétrable au moyen de boîtes de dialogue. L'interface de ce genre de logiciels est donc très souple.

2.2.Les domaines d'application des logiciels auteurs

Le domaine pour lequel la production multimédia est destinée influence également le choix du logiciel auteur utilisé. Cette section présente brièvement quelques-uns de ces domaines.

2.2.1.Les affaires

Les premières productions multimédias ont été sans aucun doute utilisées dans le domaine des affaires et plus précisément dans le marketing, la recherche et développement, la production et les resources humaines.

Une classe entière d'applications a été dédiée à ce domaine dans lequel les présentations multimédias consistent en une série de graphiques rehaussés par des animations, des commentaires et des effets sonores qui maintiennent l'intérêt de l'audience.

La plupart de ces présentations sont linéaires et peuvent être interrompues à tout moment pour permettre d'éventuelles explications supplémentaires.

2.2.2.La publicité et la promotion

L'utilisation de productions multimédias dans le domaine publicitaire est assez récent. Ce domaine qui peut se présenter sous différentes formes nécessite des types de productions multimédias allant de la simple séquence linéaire aux structures plus complexes.

Jennings (dans *Discover Windows 3.1.Multimedia, 1992, Que Corporation ,Carmel)* classe les productions multimédias du domaine de la publicité en trois catégories:

- les spots publicitaires dédicacés à un produit ou à l'amélioration de l'image d'une firme,
- les bases de données de biens et services (par exemple: une collection d'images vectorisées représentant des fournitures de bureau pouvant être utilisées dans un logiciel de gestion d'espace),
- les kioskes informationnels.

Comme pour le domaine des affaires, l'emploi de logiciels auteurs dans le domaine de la publicité s'avère prometteur.

2.2.3.La simulation et la productivité

Les firmes peuvent dès à présent stimuler la préparation et la productivité de leurs employés grâce à l'utilisation de productions multimédias.

Les sujets abordés dans ces productions concernent par exemple les mathématiques ou la comptabilité. Ils permettent une formation rapide et motivante du personnel.

Les productions de ce domaine se rapprochent très fort de celles dédiées à l'éducation. On notera toutefois la différence de public. Les unes sont utilisées par des personnes adultes alors que les autres sont destinées à des étudiants.

2.2.4.L'éducation

L'éducation représente un marché potentiel pour les productions multimédias. Celles-ci peuvent être utilisées dans le cadre des cours pour compléter la formation traditionnelle tout en développant une approche interactive. D'un point de vue psychologique, tout porte à croire à la supériorité des méthodes d'apprentissage actif par rapport aux méthodes passives.

2.2.5.Les encyclopédies et les bases de données

Les productions englobant les encyclopédies et les bases de données sont typiquement non-linéaires. Elles constituent une information indexée facilitant les recherches.

2.2.6.L'amusement (jeux)

Ces productions sont également non-linéaires. Elles exigent une interaction importante de la part de l'utilisateur. De nouveau, c'est un large marché potentiel qui s'ouvre aux productions multimédias si on se rapporte au succès des jeux d'aventure traditionnels.

3.Le développement d'une production multimédia

Le cycle de développement d'une production multimédia se rapproche très fort du cycle de développement normal d'un logiciel.

- Dans une première phase, on doit déterminer les objectifs à atteindre et délimiter le public concerné par la production.
- L'équipe nécessaire à l'élaboration d'une production multimédia est fort diversifiée. Bien que les petites productions fassent exception à cela, une équipe composée d'informaticiens, de graphistes et d'ingénieurs du son, semble indispensable pour obtenir de bons résultats.
- La phase finale: la création de la production avec période de tests et correction si nécessaires.

Voyons maintenant ce qui existe sur le marché.

4.Ce que nous propose le marché

Les logiciels présentés dans cette section ne constituent en aucun cas une liste exhaustive des logiciels auteurs mais illustrent les différents types que l'on rencontre fréquemment dans la littérature. Par ailleurs, ils permettent tous la manipulation de données multimédias.

Il s'agit de *AuthorWare Professional, Hypercard developer kit, Icon Author, Macromind Director, Multimedia Manager, Tempra Media Author, Toolbook, IBM Storyboard Live!, IBM LinkWay...*

Il est à remarquer que chacun de ces logiciels demande des configurations matérielles qui, il y a quelques mois encore, étaient non négligeables: pour les PC's, un 386 à 20Mhz et 4M en mémoire centrale semble être un minimum alors que pour la gamme des Macintosh, un MacII avec 4M est conseillé.

Pour chaque logiciel, j'essaierai de faire ressortir les points forts.

4.1.Authorware Professional

Authorware Inc., initiateur du multimédia, propose un logiciel qui a déjà fait ses preuves dans beaucoup de projets. La façon de travailler de AuthorWare Professional est très attirante: le plan de travail comporte un ensemble d'icônes permettant de réaliser l'application.

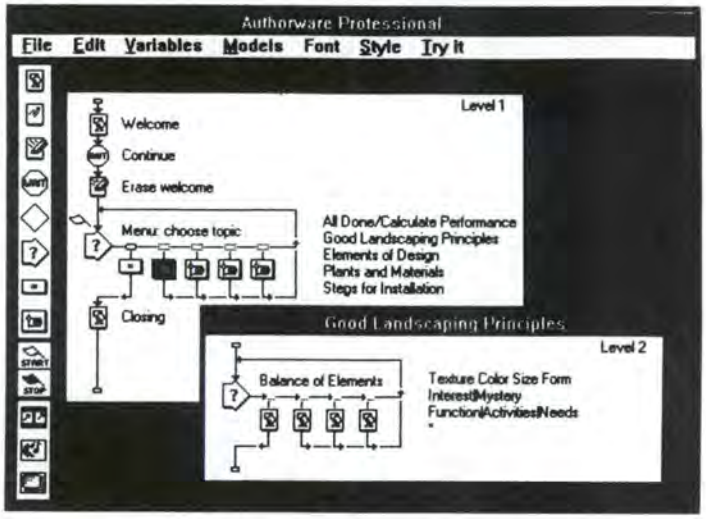


Figure 3: L'écran d'Authorware Professional.

La conception de celle-ci consiste à dessiner un diagramme du projet en plaçant différentes icônes dans la fenêtre de construction. Cette simplicité d'aspect cache pourtant des compétences remarquables et complexes.
La "boîte à outils" est composée des icônes suivantes:

- une icône d'affichage de textes et de graphiques,
- une icône d'animation servant à décrire le mouvement d'un objet à l'écran: on peut donc déterminer une trajectoire et des paramètres de contrôle du mouvement (rapidité...),
- des effets d'effacement de page ou de temporisation font l'objet de deux autres icônes,
- une icône de décision permet des branchements de différentes natures (séquentielle, aléatoire...)
- trois icônes complémentaires rendent possible la gestion du son, de l'animation et même de la vidéo...

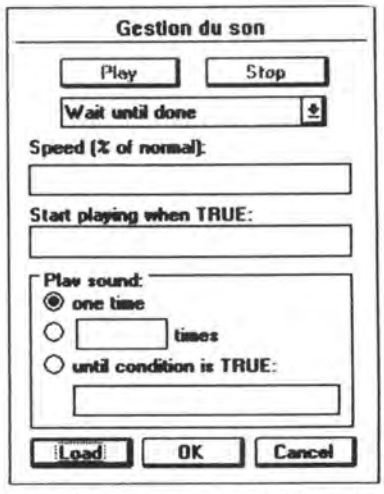


Figure 4: Boîte de dialogue associée à l'icône de gestion du son.

Authorware propose également une série de modèles. Le concepteur peut donc reprendre des "squelettes" et améliorer son efficacité puisqu'il ne doit modifier que quelques paramètres. Le système met à la disposition de l'utilisateur plus de 200 variables et fonctions qui se révèlent être alors un outil idéal pour effectuer des statistiques d'utilisation de l'application. Il peut être utilisé aussi bien sur Macintosh que sur PC.

4.2.Hypercard developer kit

Hypercard n'est pas un produit récent. Tous ceux qui sont familiers au monde de Macintosh connaissent ses possibilités. C'est lui qui a banalisé le concept d'hypertexte dans cet environnement. De plus, grâce à d'autres utilitaires comme le QuickTime, qui permet la gestion d'animation et de vidéo, le développement d'applications multimédia s'en trouve largement favorisé.
La conception d'un programme se résume à des éléments visuels que sont les fonds des cartes, les boutons etc. et à des scripts écrits en langage HyperTalk.
Une bibliothèque d'icônes sert à déclarer et à placer des boutons sur une carte .

Figure 5: L'écran d'Hypercard.

En cliquant sur ces boutons, l'utilisateur déclenchera ainsi la série d'actions associées. La définition - dans le langage HyperTalk - de ces actions, n'est

cependant pas à la portée de tout le monde et l'utilisation d'un tel logiciel reste donc réservée à des professionnels.

4.3.Icon Author

Icon Author offre au "programmeur" une prise en main rapide. Ressemblant en quelque sorte à Authorware Professional, la conception d'une application est au niveau de n'importe quel individu ayant au moins entendu parler de fichiers textes, d'images, de base de données etc. Ici également, la structure du programme à concevoir se dessine sur un organigramme et on retrouve plus ou moins les mêmes fonctionnalités que sur Authorware.

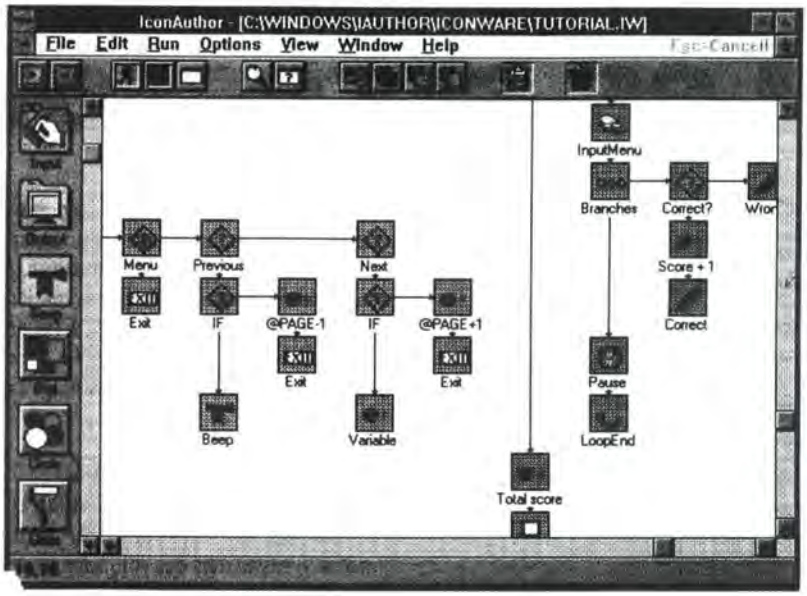


Figure 6: L'écran d'Icon Author.

La création des applications se fait grâce à six logiciels complétant ainsi l'environnement de travail:

- **SmartObject**: permet de créer des pages graphiques incluant des types de données différents (Graphes, images, objets Windows...),
- **Graphics**: est un petit éditeur de bitmaps,
- **Icon Animate**: permet l'édition d'animations,
- **RezSolution**: capture les images bitmaps,
- **Video Editor**: comme son nom l'indique, c'est un éditeur d'images vidéo,
- **IAScope**: est quant à lui un débugger.

Les applications qui sont créées par Icon Author ne sont pas compilées. Elles doivent être lancées par un interpréteur (**Present**). Celui-ci a l'avantage d'une plus grande souplesse par rapport aux différentes configurations de périphériques installés dans l'ordinateur. Icon Author tourne sous Windows.

4.4.Macromind Director

Son atout majeur est sans aucun doute son module de création d'animations particulièrement puissant. L'écran principal reprend plusieurs fenêtres qui forment un plan de montage dans lequel l'utilisateur définit les animations, les transitions et les effets spéciaux.

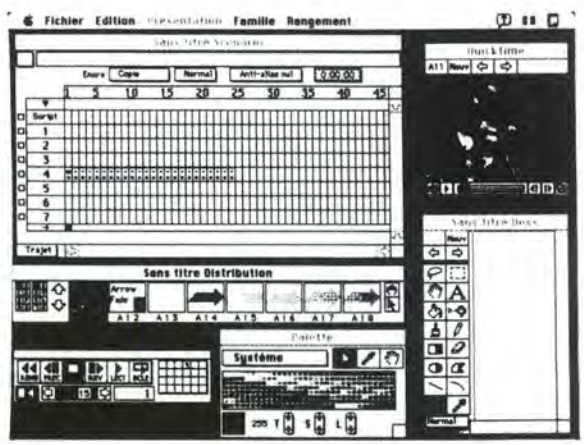


Figure 7: L'écran de Macromind Director .

On peut également ajouter des sons aux animations. Ce logiciel n'est pas à mettre entre toutes les mains. Il exige une méthode de travail rigoureuse et un certain sens artistique. Son apprentissage n'est pas aisé.

4.5.Multimedia Manager

Ce logiciel permet la conception d'un large éventail d'applications. Il s'utilise sous l'environnement Windows. Textes, images, musiques ou clips sont facilement saisis, organisés et restitués. La création d'une application consiste à concevoir des boîtes et des boutons.

Par boîtes, on entend une zone de l'écran contenant du texte ou une autre représentation visuelle.

Les boutons peuvent, quant à eux, être associées à des séquences multimédia.

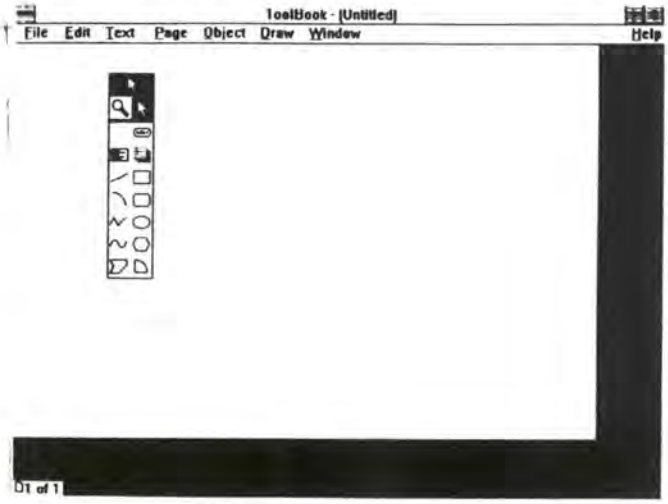


Figure 10: L'écran de Toolbook.

Dans le mode "Auteur", des outils permettent un placement facile des objets ainsi que l'affectation de certaines de leurs caractéristiques (couleurs, police de caractère, style...).
Comme pour beaucoup d'autres logiciels auteurs, Toolbook est très facile d'emploi pour de simples applications mais le développement d'applications plus sophistiquées demande des connaissances avancées de programmation.

4.8.IBM Storyboard Live!

IBM Storyboard Live! est un logiciel tournant sous DOS. Il permet de mélanger la vidéo, la voix, des animations, des dessins et du texte. L'utilisateur peut créer et organiser des présentations multimédias grâce aux cinq modules du logiciel.

- ***Electronic Presentation***: C'est le module principal puisque c'est avec lui que l'on peut concevoir la présentation. Lorsque l'on crée une présentation, ce module génère automatiquement une liste des écrans conçus dans laquelle on peut se balader afin d'éditer ou de réorganiser les différents éléments.
- ***Picture Maker***: Comme son nom l'indique, ce module permet la création et l'édition d'images utilisables dans les présentations. Il accède à une librairie de 800 images et symboles et supporte les formats de fichiers TIFF,BMP,GIF, _IM, PIC, PCX et IBM LinkWay.
- ***Story Editor***: Ce module donne accès, si on a équipé son ordinateur d'une carte vidéo, à des effets spéciaux, des animations et des sons digitalisés.

- ***Picture Taker***: Avec ce module, on peut capturer des écrans dans d'autres présentations, ou même dans d'autre programmes DOS, Windows, OS2 et de les importer dans une nouvelle présentation.
- ***Story Teller***: C'est ce module qui "exécute" les présentations.

Figure 11: L'écran de Storyboard.

Storyboard est surtout destiné aux présentations de ventes, aux démonstrations et aux présentations à caractère éducatif.

4.9.IBM LinkWay

IBM LinkWay est un programme qui permet à des non-programmeurs de développer des petites applications combinant texte, couleurs, graphiques, et images. LinkWay peut également accéder au son et à la vidéo grâce à des cartes appropriées. Il s'avère très intéressant dans le cadre scolaire où les applications peuvent aller de la simple séquence de textes et d'images à des séquences plus complexes liées à des commandes. Un langage (SCRIPT) interne au produit permet, entre autre, la manipulation des graphiques à l'écran.

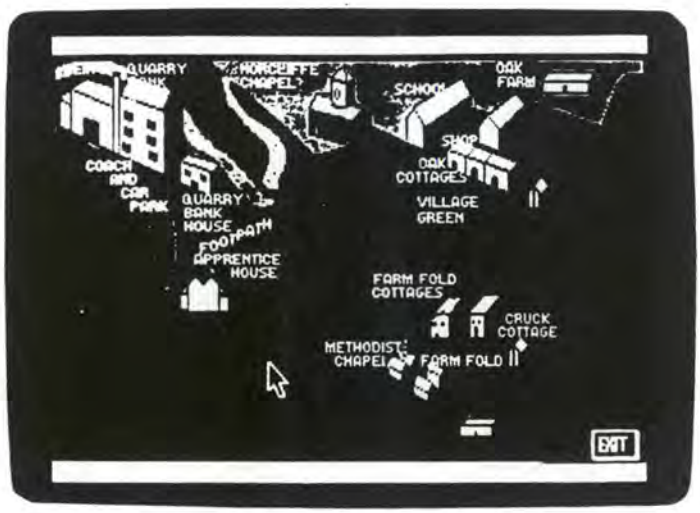


Figure 12: L'écran de LinkWay.

Plusieurs applications créées sous LinkWay accompagnent le logiciel à titre d'exemples. En les utilisant comme guides, le développement d'exercices peut s'en trouver facilité.

Les logiciels auteurs existent donc bien. Chacun d'eux a ses spécificités et est dédié à des utilisations particulières.

5.Ces logiciels conviennent-ils dans notre cas?

Pour répondre à cette question, nous allons considérer cinq critères:

Critère 1 Le temps d'apprentissage du logiciel.

Critère 2 L'étendue des applications possibles,

Critère 3 Le degré de compétence informatique exigé (connaissance d'un langage, démarche algorithmique traditionnelle, analyse de tâche, etc.).

Critère 4 Les moyens d'interaction, (souris, clavier, défilement)

Critère 5 Le coût:

Nom du logiciel	*Temps d'apprentissage*	*Etendue des applications*	*Degré de compétence informatique*	*Moyens d'interactions*	*Coût*[3]
Authorware	rapide sauf pour des réalisations plus complexes	Présentation multimédia, vente, formation	algorithmes + langage/script	classiques	~ 200.000fr
Hypercard	long pour la programmation	fort large	langage/scripts	classiques	~ 9.000fr
Icon Author	rapide sauf pour des réalisations plus complexes	Présentation multimédia, EAO	algorithmes	classiques	~140.000fr
Director	très long	animations	langage/scripts	classiques	~55.000fr
Multimedia Manager	raisonnable	dédié surtout aux sons	algorithmes	classiques	~30.000fr
Tempra Media Author	long et rigoureux	vidéo, base de données multimédia	langage	classiques + support d'écrans tactiles	~13.500fr
Toolbook	long pour des réalisations plus complexes	EAO	langage/script	classiques	~45000fr
Linkway	rapide sauf pour des réalisations plus complexes	EAO	algorithmes + langage/script	classiques	???
StoryBoard	relativement rapide	présentations multimédias, vente, EAO	algorithmes	classiques	???

Tableau 1: Récapitulatif .

De ce tableau, nous pouvons dégager plusieurs remarques:

[3] *Les prix des logiciels ont été tirés de "Logiciels auteurs" in Multimedia,,Gilles Fouchard, Mars 93. Ils sont approximatifs et sont susceptibles d'être modifiés.*

- Dans la plupart des cas, le temps nécessaire pour maîtriser le logiciel est loin d'être négligeable. Cela est lié au degré de compétence informatique requis (lenteur de l'apprentissage d'un langage). Seuls deux logiciels se dégagent. Ils accusent toutefois une faiblesse quant à la diversité et la complexité de leurs productions.

- Les moyens d'interaction des logiciels ne sont que difficilement adaptables pour des personnes présentant des déficiences physiques. Pratiquement tous utilisent les moyens classiques tels que le clavier et la souris. Seul le **Tempra Media Author** propose un support d'écrans tactiles. Bien entendu, le prix de la configuration matérielle monte alors en flèche.

- La dernière remarque concerne le prix de ces logiciels. Certains d'entre eux dépassent de loin les limites acceptables pour les budgets des institutions pour personnes handicapées.

Comme nous pouvons le voir, les logiciels de cette liste ne répondent pas vraiment aux exigences du domaine du handicap. L'élaboration d'un nouveau logiciel paraît dès lors pertinente.

6.Proposition d'un logiciel adapté

5.1.Les principes de base

On peut diviser les caractéristiques du logiciel selon deux niveaux:

Le premier niveau concerne les caractéristiques des ***exercices destinés aux personnes handicapées mentales***.

- Les exercices doivent être basés sur des images et des sons.
- Les écrans doivent rester simples et comporter le moins d'éléments possible.
- Les manipulations durant l'exécution des exercices doivent être réduites au maximum et, dans la mesure du possible, simplifiées.
- Les exercices doivent pouvoir s'adapter à la population et doivent donc être paramétrables au point de vue des moyens d'interaction et des caractéristiques

des images (tailles, couleurs,...). Le niveau de difficulté devrait également être adaptable.

Le second niveau est, quant à lui, destiné au ***logiciel auteur proprement dit***.

- Il doit être facile d'emploi.
- Il doit permettre la paramétrisation des images et des moyens d'interaction.
- Il doit pouvoir enregistrer tous les événements qui ont eu lieu durant une session d'exercice. C'est ce qu'on appelle la *trace* de l'exécution. Grâce à elle, les erreurs de manipulation (par exemple cliquer en dehors d'un objet interactif) ainsi le temps écoulé entre deux actions consécutives peuvent donner des indications sur le niveau de compréhension atteint par l'utilisateur.

Si nous reprenons les critères du point 5, nous pouvons faire ressortir les maîtres-mots suivants:

- la **diversité** des exercices que l'on peut concevoir (cfr. critère 2),
- l'**interface** appropriée au domaine du handicap.(cfr. critère 4),
- la **simplicité** d'utilisation lors de la conception et de l'exécution des exercices. (cfr. critères 1 et 3).

Dans ***Auteur!***, la **diversité** des exercices est assurée par les différents outils mis à la disposition du concepteur. Ceux-ci reprennent les outils développés l'an passé auxquels on a ajouté le texte, l'animation et le son. Associés à ces outils, les actions apportent une dynamique à l'application conçue.

Le deuxième point concerne l'**interface**. La population ciblée dans ce logiciel demande un soin tout particulier dans l'élaboration de celle-ci.

La **simplicité** est bien sûr de rigueur. Les utilisateurs ne peuvent pas se permettre de consacrer énormément de temps à l'apprentissage du logiciel et à "l'implémentation" des exercices. ***Auteur!*** doit donc permettre une prise en main rapide et aisée.

Comme on peut le remarquer, les premier et troisième points font l'objet d'un compromis: permettre la diversité des exercices sans alourdir la simplicité d'utilisation.

5.2.La diversité

5.2.1.Les outils disponibles

Les premiers outils sont ceux qui ont été développé l'an passé. Une brève description en a été donnée au chapitre 2. Le lecteur intéressé par l'analyse se rapportant à ceux-ci pourra consulter le mémoire de Stéphanie Baudrenghien et de Rudy Démo [DEMO,92].

Il s'agit :

- de la liste fixe,
- du dérouleur,
- de la boîte de dialogue,
- de la boîte d'édition,
- de l'écran,
- de l'icône.

A ceux-ci, ajoutons encore:

- **L'animation**:

Cet outil, tout comme le suivant, prend tout son intérêt dans l'utilisation de renforçateurs. L'animation sera composée d'une série de pictogrammes. Le concepteur de l'application pourra définir plusieurs paramètres tels que les différents pictogrammes composant l'animation, la vitesse de défilement, la position à l'écran ainsi que les dimensions de l'animation.

- **Le son**:

Avec l'arrivée de Windows 3.1, la gestion des sons est rendue possible. Le fait d'inclure cette possibilité dans la boîte à outils ne peut que l'enrichir et sera abondamment utilisée dans la conception des exercices. Les paramètres à considérer seront le nom du fichier son et le nombre de fois qu'il faudra le jouer.

- **Le texte**:

Cet outil, quant à lui, servira plus au concepteur des exercices. Lorsque les exercices élaborés deviennent complexes (plusieurs écrans et niveaux de menu), il peut alors être intéressant pour le moniteur de se repérer grâce à un commentaire. Cet outil permettra d'éditer facilement ces commentaires.

L'analyse conceptuelle de ces trois outils fait l'objet du chapitre suivant.

5.2.2.Les actions

A chaque outil visuel, le concepteur des exercices pourra associer différentes actions. Ces actions seront en fait le "moteur" des exercices conçus. Lors de l'exécution, un objet sélectionné par la personne handicapée déclenchera la série d'actions qui lui est associée.

On peut regrouper les différentes actions en plusieurs catégories:

- **les actions sur les objets**

Pour la gestion des actions sur les objets, un système d'identifiants a été mis en place. Cet identifiant, composé d'une série de caractères alpha-numériques, fera désormais partie de la liste des paramètres de chaque objet.

Chaque action sera définie par son nom et par la série de paramètres propres à l'action (cfr Tableau 2).

NOM	*PARAMETRES*
-l'apparition (*)	L'id. de l'objet, les paramètres spécifiques à l'objet (cfr chapitre 2).
-la disparition	L'id. de l'objet.
-la modification (*)	L'id. de l'objet, les paramètres spécifiques à l'objet (cfr. chapitre 2).
-le positionnement	L'id. de l'objet, la position sur l'écran (ordonnée et abscisse en pixels)
-le déplacement	L'id. de l'objet, la position de destination sur l'écran (ordonnée et abscisse en pixels)
-l'exécution (animation- son)	-*pour l'animation*: la liste des pictogrammes de l'animation, la vitesse de défilement, la position de destination (abscisses et ordonnées en pixels) -*pour le son*: le nom du fichier son ainsi que le nombre de fois qu'il doit être joué

Tableau 2: Les actions sur les objets.

Il est clair que l'objet qui a déclenché l'action et celui sur lequel l'action est exécutée ne doivent pas être obligatoirement les mêmes.
Les actions marquées par (*) sont des actions 'génériques'. En fait, pour chaque type d'objet, correspond l'action spécifique correspondante. Par exemple, l'appel de l'écran dont l'identifiant est 'MENU' se traduira par l'action apparition dont le paramètre unique sera l'identifiant 'MENU'.

- **les actions sur des variables (opérations)**

Les actions précédentes concernent la manipulation des objets en eux-mêmes. Dans l'analyse des différents logiciels élaborés précédemment, on remarque que ceux-ci contiennent une série de traitements. Comme ces traitements s'opèrent généralement sur des variables, le logiciel auteur devra donc permettre la création et la manipulation de celles-ci.

Les actions suivantes sont donc souhaitables:

-création d'une variable,
-assignation d'une valeur à une variable,
-addition et soustraction de deux expressions,

-multiplication et division de deux expressions,
-tests d'égalité, du plus petit, du plus grand.

- Il serait intéressant de pouvoir faire des **traitements élémentaires** sur ces actions, à savoir:

-la répétition d'une séquence d'action,
-la possibilité de faire des branchements conditionnels (correspond au "si...alors...sinon").

5.3.L'interface spécifique au handicap

La conception d'une interface destinée à des personnes handicapées demande une étude appronfondie. La diversité des capacités *mentales et physiques* de la population concernée ne peut que confirmer la souplesse d'adaptation que doit revêtir l'interface.
Cette adaptation doit se faire au niveau du contenu des outils ainsi qu'au niveau des moyens d'interactions avec le logiciel.

5.3.1.Les outils

Les outils proposés ont spécialement été conçus pour être utilisés par des personnes handicapées mentales. Ils ont fait l'objet d'une évaluation. Les caractéristiques principales de ces outils peuvent être paramétrées par l'éducateur: couleurs significatives, tailles adéquates, images pertinentes, etc.

La figure suivante montre la barre d'outils utilisée dans ***Auteur!*** .

boutons déclenchant le chargement et la sauvegarde des exercices

bouton associé à l'outil "écran"

bouton associé à l'outil "icône"

bouton associé à l'outil "animation"

bouton associé à l'outil "texte"

bouton associé à l'outil "dérouleur"

bouton associé à l'outil "boîte d'édition"

Figure 13: La barre d'outils d'***Auteur!***.

La barre d'outil permettra la sélection des outils par un simple clic de souris.

5.3.2.L'accès

Les moyens d'interaction avec le logiciel sont très importants. Ils permettent l'adaptation de personnes possédant des capacités physiques ou intellectuelles fort diverses.

Il existe beaucoup de moyens d'interaction. Je ne reprendrai ici que ceux qui sont susceptibles d'être retenus dans ***Auteur!*** . Ce choix est dicté par des impératifs de minimisation de coûts. Ce problème dont on a encore peu parlé est pourtant fort présent. En effet, les institutions pour lesquelles est destiné le logiciel auteur ne peuvent pas se permettre d'investir des sommes considérables dans un matériel coûteux.

Le moyen le plus utilisé est certainement la **souris**. Cependant, dans bien des cas, les utilisateurs éprouvent des difficultés à la manipuler . En effet, elle requiert une grande précision dans les mouvements. Elle ne sera donc pas utilisable par tout le monde.

La **trackball** est, quant à elle, une sorte de souris renversée. Le socle ne bouge pas et les déplacements du pointeur se font selon le même principe que la souris. Ce système est pratique car il permet de décomposer les manipulations: le déplacement et la validation.

Le **joystick** est lui aussi très connu. Celui-ci permet d'indiquer la direction que doit suivre le pointeur.

Pour ces trois moyens d'interaction, le concepteur devrait pouvoir déterminer la vitesse de déplacement du pointeur pour affiner une fois de plus l'adaptation de chaque utilisateur.

En plus de ces moyens viennent s'ajouter les mécanismes d'**interrupteurs**.
Ces mécanismes permettent la sélection et la validation des objets situés à l'écran grâce à quelques boutons. Les objets qui peuvent être sélectionnés se trouveront chacun à leur tour mis en évidence et la validation se fera sur cet objet.

En jouant sur le nombre de ces boutons, on peut également adapter l'interface en fonction des capacités de l'utilisateur.

- Un bouton: On doit utiliser des méthodes de défilement automatique. Le bouton sert alors à la validation de l'objet.
- Deux boutons: Le premier bouton sert à la sélection des objets. Il contrôle donc le déplacement . Le second sert à la validation.
- Trois boutons: Deux boutons peuvent être utilisés pour les déplacements (horizontaux et verticaux) et le troisième pour la validation.

Pour tous ces moyens d'interaction, il est important de pouvoir filtrer les entrées afin d'éviter les sélections ou validations accidentelles.

On peut retenir plusieurs paramètres qui permettront de filtrer ces entrées:

- **"Debounce time"**: C'est le temps qui doit s'écouler entre le changement d'état d'un bouton et le moment où l'ordinateur est prêt à accepter le prochain changement.

- **"Input acceptance time"**: Cela permet de rejeter les actions involontaires de l'utilisateur. Il spécifie le temps minimum pendant lequel un bouton doit être enfoncé pour que l'ordinateur tienne compte de l'action.

- **"Post acceptance delay"** : Cela détermine le temps après lequel l'ordinateur pourra tenir compte de la prochaine action de l'utilisateur.

- **"Edge triggering"** : Ce paramètre détermine si l'action doit être prise en compte par l'ordinateur lorsque l'utilisateur enfonce le bouton ou quand il le relâche.

- **"Repeat functions"** : Pour permettre qu'une sélection répétitive se fasse automatiquement, il faut définir deux autres paramètres. Le premier est le "Repeat Acceptance Delay" qui indique le temps pendant lequel un bouton doit être maintenu enfoncé pour déclencher ces fonction de répétition. Le second est le "Repeat Time" qui détermine le temps entre deux répétitions.

Ces paramètres ont été définis par David Colven de ACE Centre dans le document "A Common Terminology for Switch Controlled Software".

La figure suivante montre comment ***Auteur!*** permet le contrôle des moyens d'interaction:

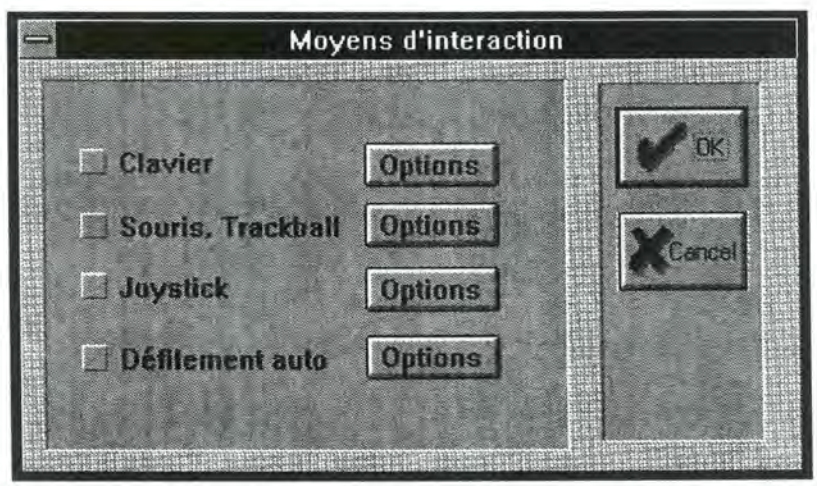


Figure 14: La boîte de dialogue contrôlant les moyens d'interaction du logiciel.

Il suffit de cliquer sur le bouton "options" associé au moyen d'interaction afin de faire apparaître la boîte de dialogue définissant les paramètres spécifiques à chaque interface.

L'exemple de la figure suivante permet le paramétrage du comportement de la souris.

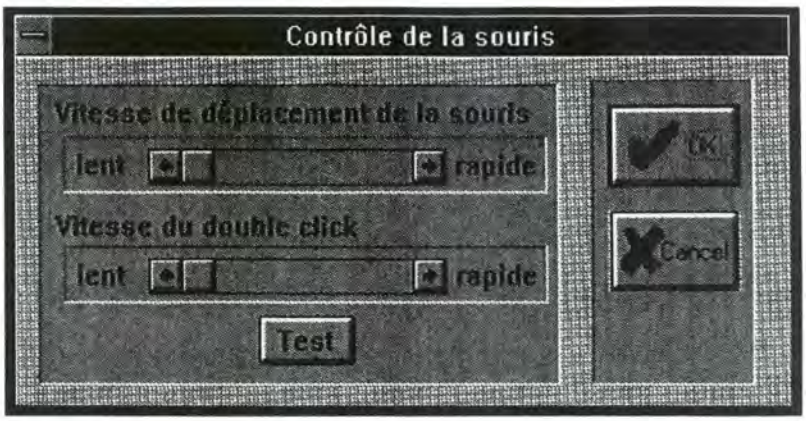


Figure 15: La boîte de dialogue contrôlant les paramètres de la souris.

5.3.3.La trace de l'exercice

J'ai déjà soulevé à plusieurs reprises le problème de la diversité des profils des personnes présentant des déficiences. Il serait intéressant de pouvoir enregistrer quelques renseignements généraux tels que le nom, l'âge, etc afin d'établir un mini-dossier sur chaque enfant. Lorsque l'éducateur commencera une séance, il identifiera l'enfant et déclenchera un mécanisme de trace permettant le relevé des événements de la session. Ces observations pourraient, par la suite, faire l'objet d'une analyse.

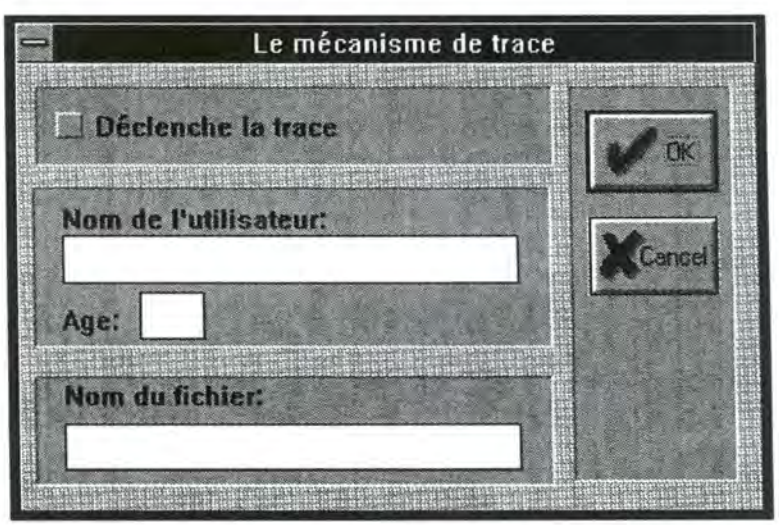


Figure 16: La boîte de dialogue contrôlant le mécanisme de trace.

La boîte de dialogue représentée à la figure précédente permet de spécifier si le mécanisme de trace doit être actionné ou non. Elle donne également la possibilité d'entrer le nom et l'âge de l'enfant qui va utiliser l'exercice et le nom du fichier dans lequel la trace devra être enregistrée.

La figure suivante donne un exemple de fichier trace:

```
------------------------------------------------------------------------------
Utilisateur: GUILLAUME
Date: 24/08/93     Heure: 09:04:26
------------------------------------------------------------------------------
Mode d'utilisation: souris
------------------------------------------------------------------------------
0m02s32c: clique sur icône "orange" (425,311) dans écran "exercice-couleurs"
0m04s17c: clique sur icône "ok" (326,212) dans écran "mauvais"
0m07s54c: clique sur écran "exercice-couleurs" (312,376)
                              :
                              :
                              :
```

Figure 17: Fragment d'un fichier de trace.

5.4.La simplicité

5.4.1.Les deux modes : la conception et l'exécution

Dans les logiciels auteurs, on rencontre deux niveaux d'utilisation.

Le premier niveau concerne la **conception** des exercices. L'utilisateur du logiciel est alors l'éducateur. Tous les outils sont mis à sa disposition pour créer les différents écrans, les objets qui les composent ainsi que les actions s'y rapportant. L'éducateur peut également choisir les moyens d'interaction s'adaptant au mieux à la personne pour laquelle l'exercice est développé. Lorsque la conception de l'exercice est terminée, le logiciel peut passer en mode **exécution** et l'éducateur doit céder sa place. Les outils ayant servi à la conception disparaissent de l'écran qui présente maintenant le début de l'exercice. Il convient de pouvoir passer d'un mode à l'autre de façon aisée pour permettre l'implémentation, le test et la correction d'un exercice de manière interactive et rapide.

Figure 18: Le menu permettant de basculer dans le mode "Exécution".

Basculer en mode "Exécution" peut se faire grâce au menu (cfr. figure précédente). L'opération inverse doit se faire, quant à elle, par la combinaison de touche ALT + C[4] puisque, dans ce cas-là, le menu est invisible .

5.4.2.Les sources

Un problème auquel sont confrontés les concepteurs des exercices se situe au niveau des graphiques. En effet, ceux-ci doivent être choisi avec un soin tout particulier et demandent en général la collaboration d'un dessinateur.

Une solution, ou du moins un début de solution est, semble-t-il, de permettre une compatibilité avec d'autres logiciels graphiques qui possèdent en général une large bibliothèque de dessins. Ainsi, le logiciel auteur reconnaîtra le format de fichier BMP qui est le standard Windows en matière de dessins "bitmaps".

Le même genre de problème, causé par l'introduction des sons dans le logiciel auteur, a été atténué par le choix d'un format de fichier audio standard: les fichiers WAV.

5.4.3.La création des écrans

5.4.3.1La structure

Les outils mis à la disposition du concepteur d'exercices seront présentés sous forme d'icones situées sur le côté gauche de l'écran. L'éducateur pourra ainsi placer les différents éléments de l'écran à créer de manière directe grâce à la souris. Cette première phase constitue la charpente de l'exercice.

[4] *C pour "Conception".*

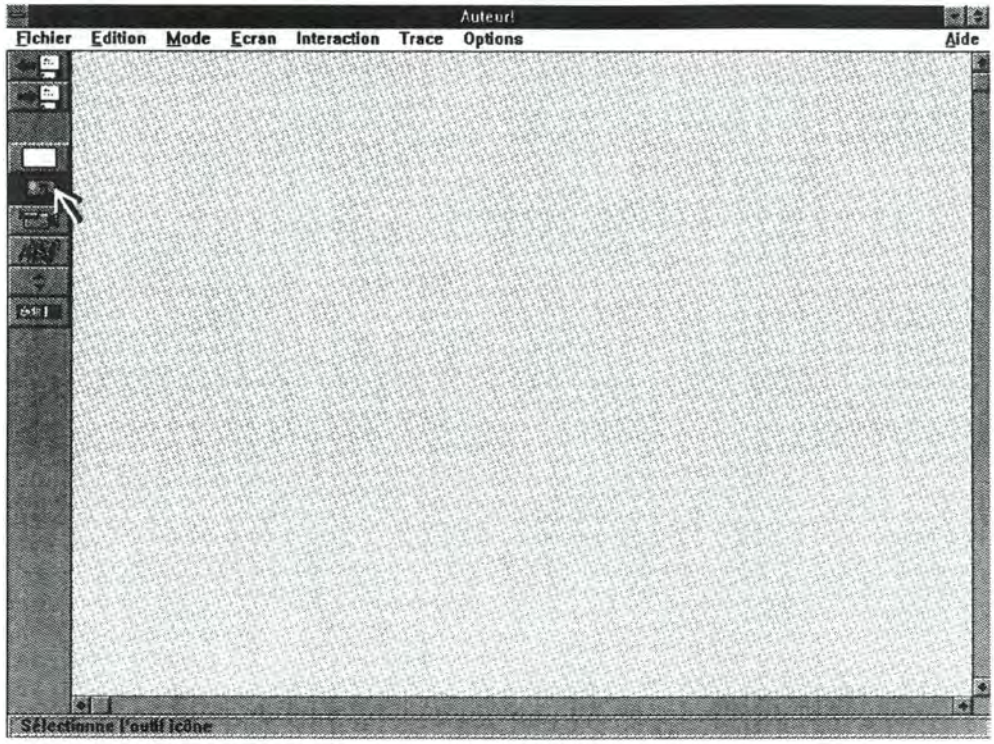


Figure 19: Etape 1: Sélectionner l'outil dans la boîte à outil.

Lorsque l'outil a été sélectionné, il suffit de le placer sur l'écran en cliquant à l'endroit désiré.

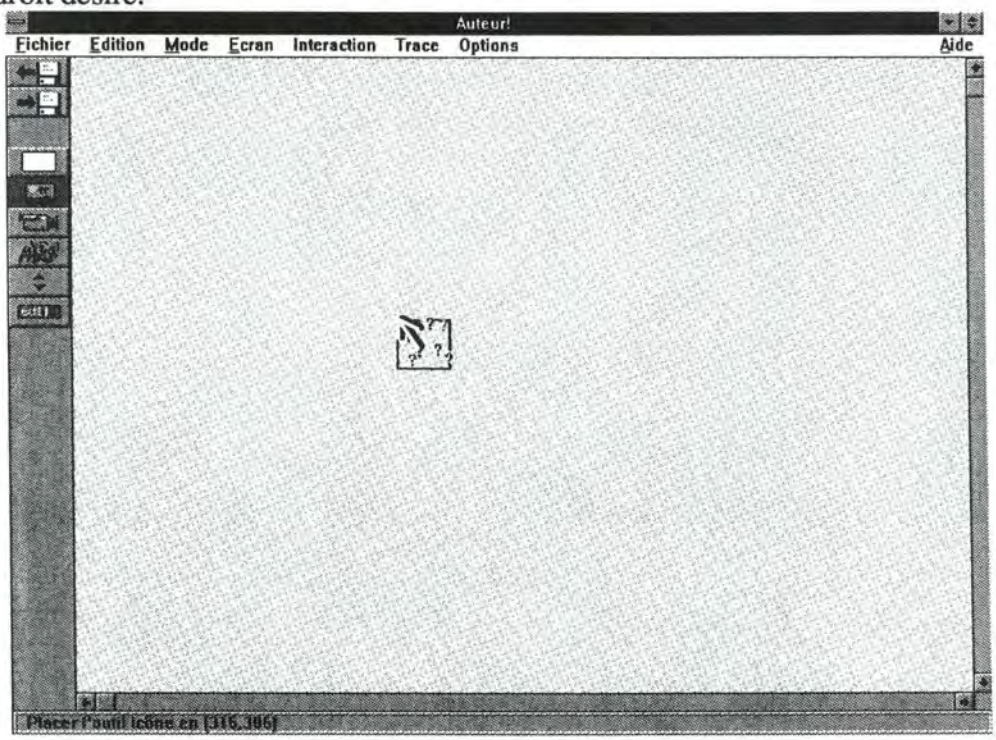


Figure 20: Etape 2: Placer l'objet sélectionné sur l'écran.

5.4.3.2.Le paramétrage

Il reste maintenent à affiner l'exercice en paramétrant les différents éléments. Cela se fait en cliquant deux fois sur un objet. Une boîte de dialogue s'ouvre et permet de rentrer les caractéristiques de l'élément.

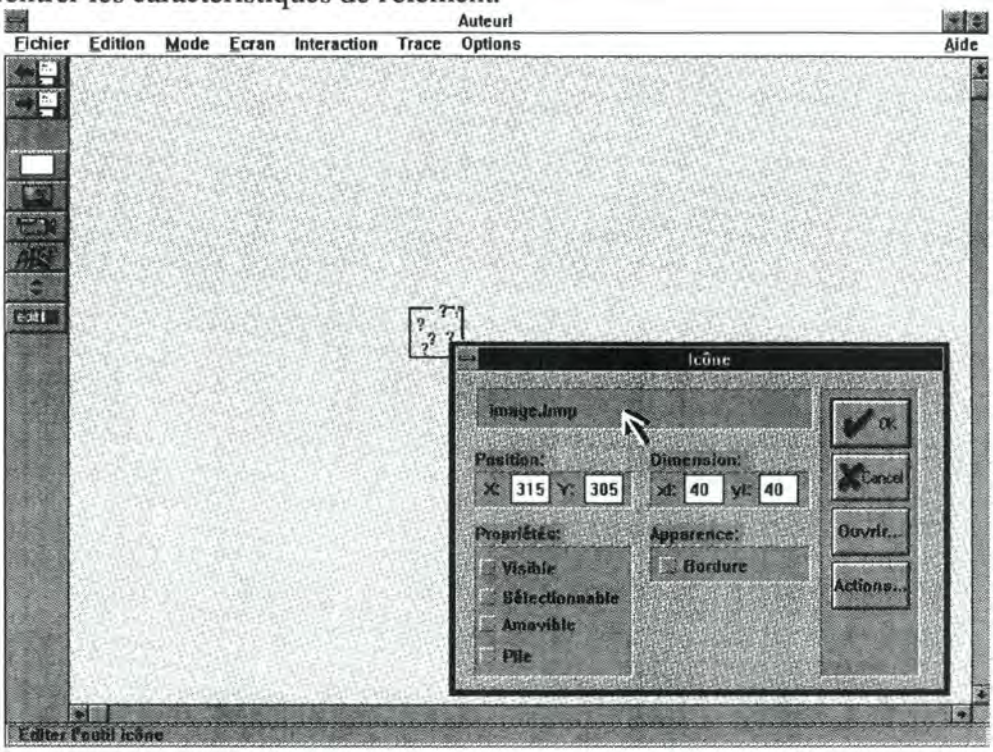


Figure 21: Etape 3: Cliquer sur l'objet fait apparaître la boîte de dialogue permettant sa paramétrisation.

C'est également à ce moment que le concepteur peut déterminer les actions à associer à l'objet qu'il est en train d'éditer. Chaque action est associée à une boîte de dialogue selon le même principe que les outils.

Conclusion

Dans ce chapitre, nous avons effectué un tour d'horizon des logiciels auteurs existant sur le marché en essayant d'en faire ressortir, à chaque fois, les caractéristiques.

Nous avons également dégagé les caractéristiques principales que devrait posséder un logiciel auteur destiné au domaine du handicap.

La confrontation des caractéristiques des logiciels du marché a dévoilé certaines lacunes et a renforcé l'idée de développer un nouveau logiciel: ***Auteur!*** .

Grâce à ***Auteur!***, l'implémentation d'exercices se trouve *facilitée* et peut se faire *rapidement*. Toutefois, cela ne signifie pas qu'il ne reste plus rien à faire. La *conception* d'un exercice demandera toujours autant de soin et de temps de la part de l'éducateur. Cette phase, qui constitue le cahier des charges de chaque exercice, est indispensable pour obtenir des résultats pertinents.

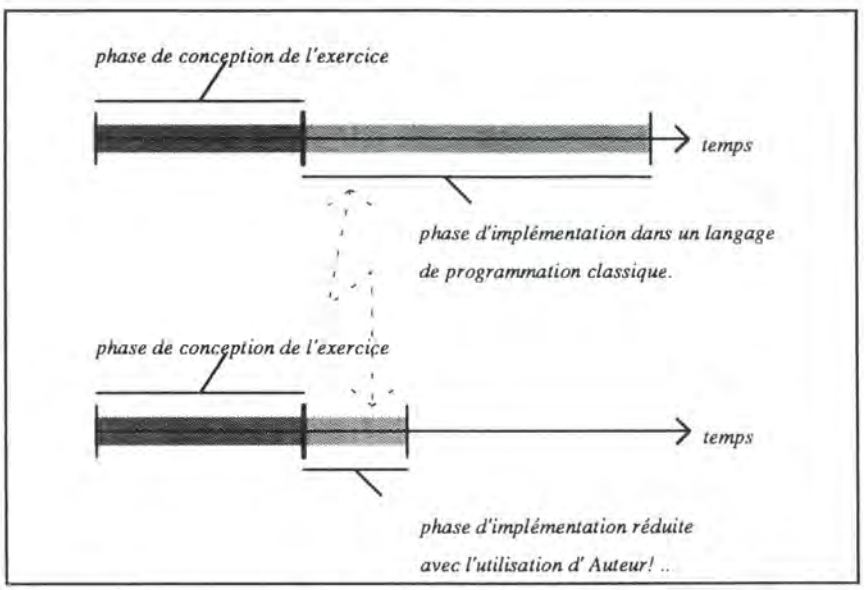


Figure 22: L'utilisation d'***Auteur!*** diminue le temps nécessaire à l'implémentation d'un exercice.

Chapitre 4

La conception

L'analyse conceptuelle des nouveaux objets introduits dans la boîte à outils est décrite dans ce chapitre. Cette description a été faite en OBLOG pour assurer une certaine homogénéïté par rapport à ce qui a été fait l'an passé.

1.La méthode utilisée

L'approche suivie pour la modélisation du logiciel auteur repose sur une analyse **orientée objet**. Cette approche est particulièrement adéquate compte tenu des raisons suivantes:

- Une approche orientée objet est l'une des approches permettant la conception d'une architecture flexible et décentralisée. Une telle architecture facilite les modifications et les extensions à apporter durant le cycle de développement de l'application. Ces qualités sont plus que jamais nécessaires dans le cadre du projet qui nous concerne puisque celui-ci a été étallé sur plusieurs années et a requis quelques aménagements de la boîte à outils.

- Par ailleurs, les exercices réalisés grâce à la boîte à outils sont composés d'**objets** graphiques attachés à des **actions**. Cette structuration centrée autour les données plutôt qu'autour des traitements a également influencé le choix d'une méthode orientée objet.

- Finalement, l'environnement Windows dans lequel l'application a été développée utilise de manière interne des concepts orientés objets (par exemple pour le fenêtrage) et, par conséquent, encourage l'utilisation de telles méthodes.

Pour assurer une certaine homogénéïté par rapport au travail effectué l'an passé, les éléments ajoutés dans la boîte à outils (les animations, le son et le texte) seront spécifiés dans le même langage de spécification utilisé dans [DEMO,92]: le langage OBLOG.

2.OBLOG

Le but de cette section est de présenter brièvement les quelques concepts d'OBLOG nécessaires pour la bonne compréhension des schémas qui vont suivre. Le lecteur intéressé trouvera de plus amples renseignements dans [DEMO,92] et [ZEIP,92].

OBLOG est un atelier logiciel, développé à l'ESDI, dont l'objectif final est de permettre l'utilisation d'un langage de spécification unique, formel, orienté objet et pictographique à travers tout le cycle de développement d'une application.

Le caractère graphique du langage simplifie son utilisation et le rend relativement intuitif.

2.1.Les objets et les classes d'objets

Un objet contient des attributs et des événements. Il est caractérisé par son comportement et par son état. L'état d'un objet est représenté par les valeurs de ses attributs à un moment donné.

Une classe d'objets regroupe l'ensemble des objets "similaires" appelés instances de la classe.

On décrit la classe par:

- son diagramme matriciel,
- le diagramme de son comportement,
- les initialisations des valeurs des attributs (ainsi que les mises à jour de ces derniers),
- les interactions avec les autres classes.

La figure 1 montre un exemple de représentation graphique de classes. Le "1" et le "?" indiquent respectivement que la classe peut contenir une ou plusieurs instances.

Figure 1: Représentation graphique de classes.

2.2.Les attributs

Les attributs des classes prennent leurs valeurs dans les domaines définis par les types de données (prédéfinis ou définis par le concepteur) ou les classes d'objets.

Un attribut peut posséder plusieurs caractéristiques:

Il peut être *constant* ou *variable*. Il est *constant* si sa valeur est fixée dès la naissance de l'objet et n'est jamais modifiée durant son cycle de vie. Dans le cas contraire, il est *variable*.

Un attribut peut être *total* ou *partiel*. Il est *total* lorsqu'il doit avoir une valeur à tout moment durant le cycle de vie d'un objet. S'il peut être indéfini à certains moments du cycle de vie, il est *partiel*. Remarquons qu'un attribut *partiel* est d'office *variable*.

Certains attributs sont utilisés pour identifier les différentes instances de la classe. Ils sont alors appelés *attributs clefs*.

La figure suivante, tirée directement de [DEMO,92], montre la description d'un objet représentant un carré.

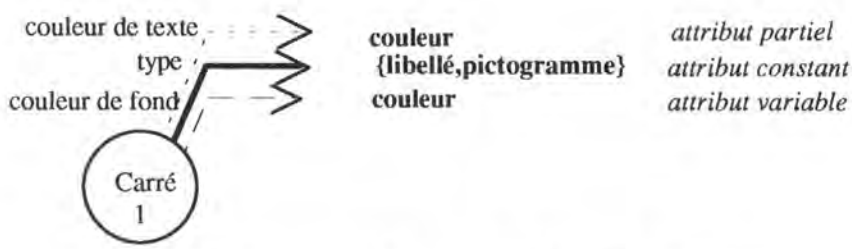


Figure 2: Représentation graphique des attributs d'une classe

Source: [DEMO,92], p. 48.

"*...Un carré peut être illustré soit par un libellé, soit par un pictogramme mais lorsqu'il est créé, il ne peut plus changer de type. S'il est un pictogramme, il n'aura évidemment pas de couleur de texte. Cet objet peut en outre changer de couleur de fond pendant sa vie...*" ***Source***: [DEMO,92], p. 48.

2.3.L'événement

Un événement est, par définition, une action atomique qui affecte la vie et/ou l'état d'un objet.

La vie d'un objet commence par un événement de naissance (marqué par "*"), il est suivi d'une série d'événements de mise à jour et se termine par un événement de mort (marqué par "+").

Notons encore qu'un événement d'un objet peut être *actif* s'il est déclenché par cet objet ou *passif* dans le cas contraire. Un événement *actif* est marqué par "!". Si un événement est réalisé par l'"extérieur", il est marqué par "o".

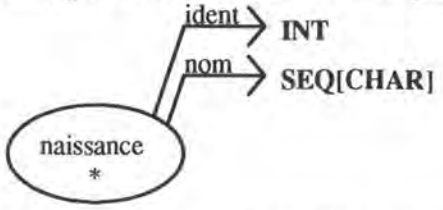


Figure 3: Représentation graphique d'un événement.

La figure précédente représente la naissance d'un objet auquel seront attachés un nom (alphanumérique) et un identifiant (numérique).

2.4.La description d'un objet

Cette description contient la définition des attributs et des événements des objets constituant la classe. Elle est représentée par ce qu'on appelle le *diagramme matriciel* de la classe.

2.5.Le comportement d'un objet

Le comportement d'un objet est constitué des séquences d'événements possibles constituant le cycle de vie de l'objet. Le comportement des objets est représenté par le diagramme de comportement de leur classe.

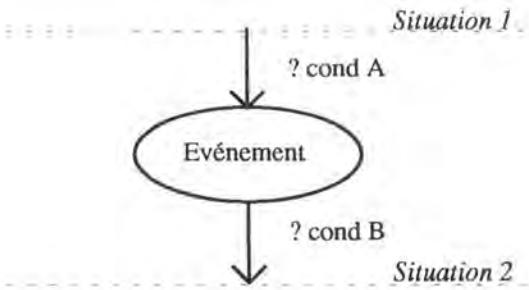


Figure 4: Représentation graphique d'un fragment d'un diagramme de comportement.

Des contraintes peuvent être définies pour déclencher les événements. Dans l'exemple de la figure précédente, l'événement ne se déclenchera que lorsque la précondition A sera remplie et lorsque la situation 2 vérifiera la condition B.

La description de ces quelques concepts devraient suffire pour la compréhension des schémas suivants. Rappelons toutefois que le lecteur intéressé trouvera des renseignements supplémentaires sur OBLOG dans [DEMO,92] et [ZEIP,92].

3.Description des nouveaux outils

La description des nouveaux outils demande une déclaration préalable de plusieurs types de données non prédéfinis.

- POINT est un type de données représentant les coordonnées d'un point de l'écran. Il est défini comme suit:

```
POINT : record [
        INT: x
        INT: y
        ]
```

- RECT représente une zone rectangulaire à l'écran. Il est défini par:

```
RECT : record [
        POINT : ptsupgauche
        POINT : ptinfdroit
        ]
```

- PICTOGRAMME est un type de donnée représentant une image à l'écran. Il est défini par:

```
PICTOGRAMME:
    record [
        STRING: nompictogramme
        Seq[INT]: image
        ]
```

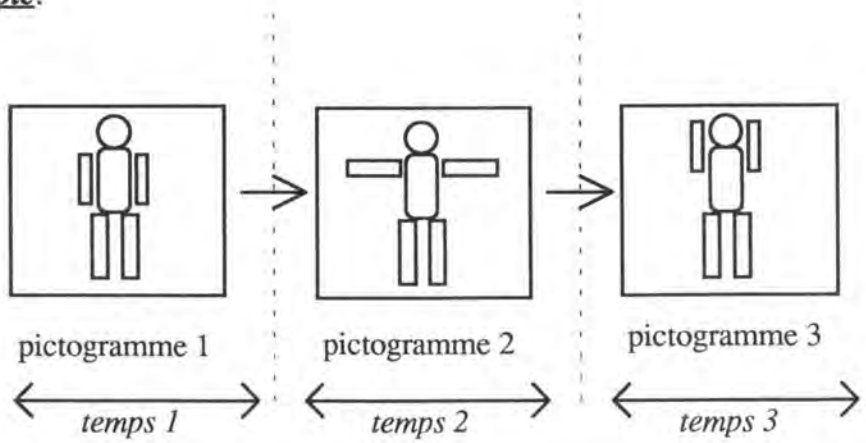
- POLICE donne la description d'une police de caractère:

```
POLICE:
    record [
        STRING: nompolice
        INT: taille
        BOOL: italique
        BOOL: souligné
        BOOL: gras
        ]
```

3.1.L'animation

Définition: L'animation est un outil proche de l'icône. Elle est composée d'une série de pictogrammes qui sont affichés , un à un, à l'écran selon la vitesse de défilement choisie.

Exemple:



Les trois pictogrammes affichés les uns après les autres donnent l'effet d'animation d'un homme levant les bras.

Cet outil sera surtout utilisé pour les renforçateurs.

Diagramme matriciel:

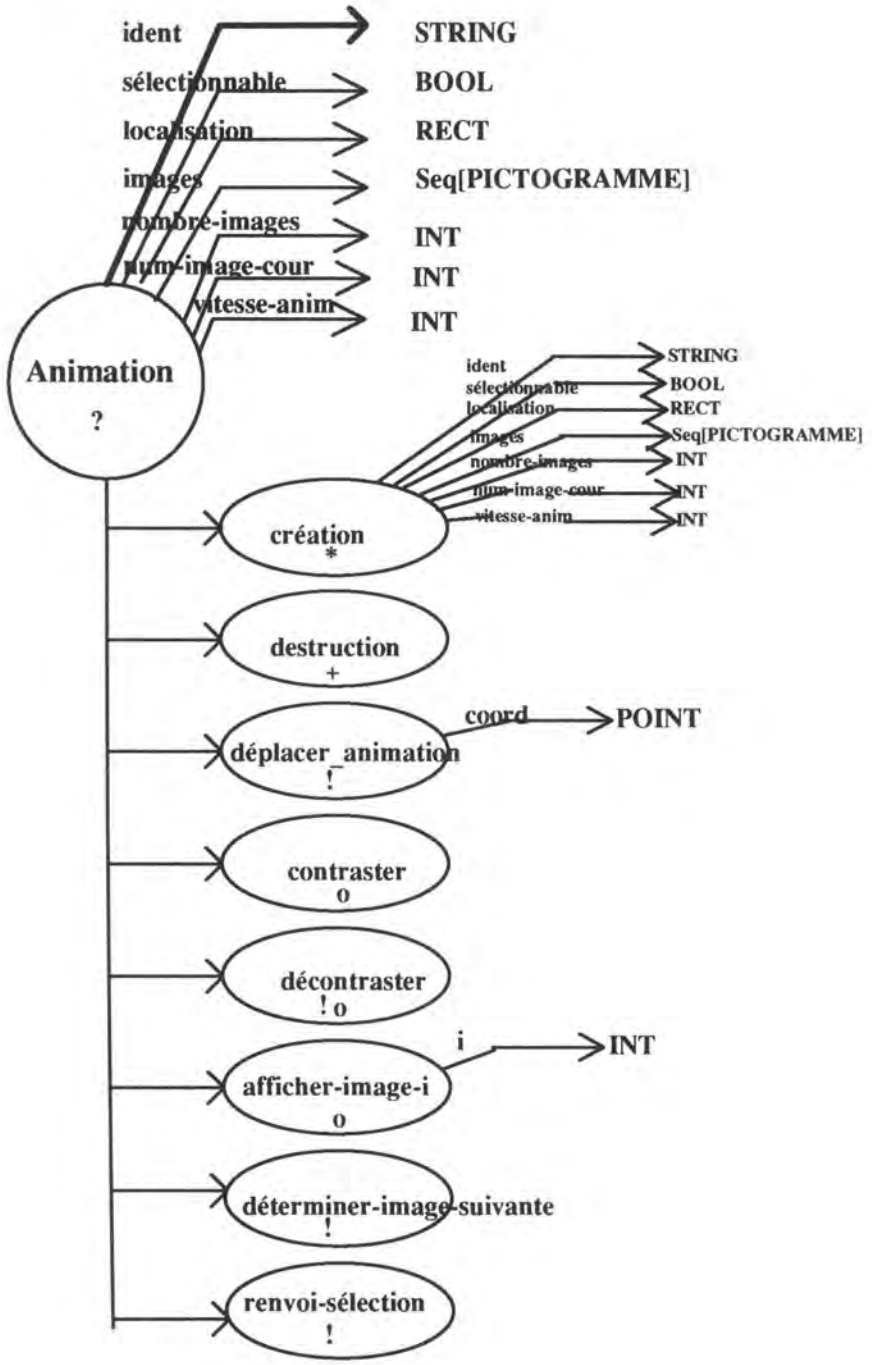
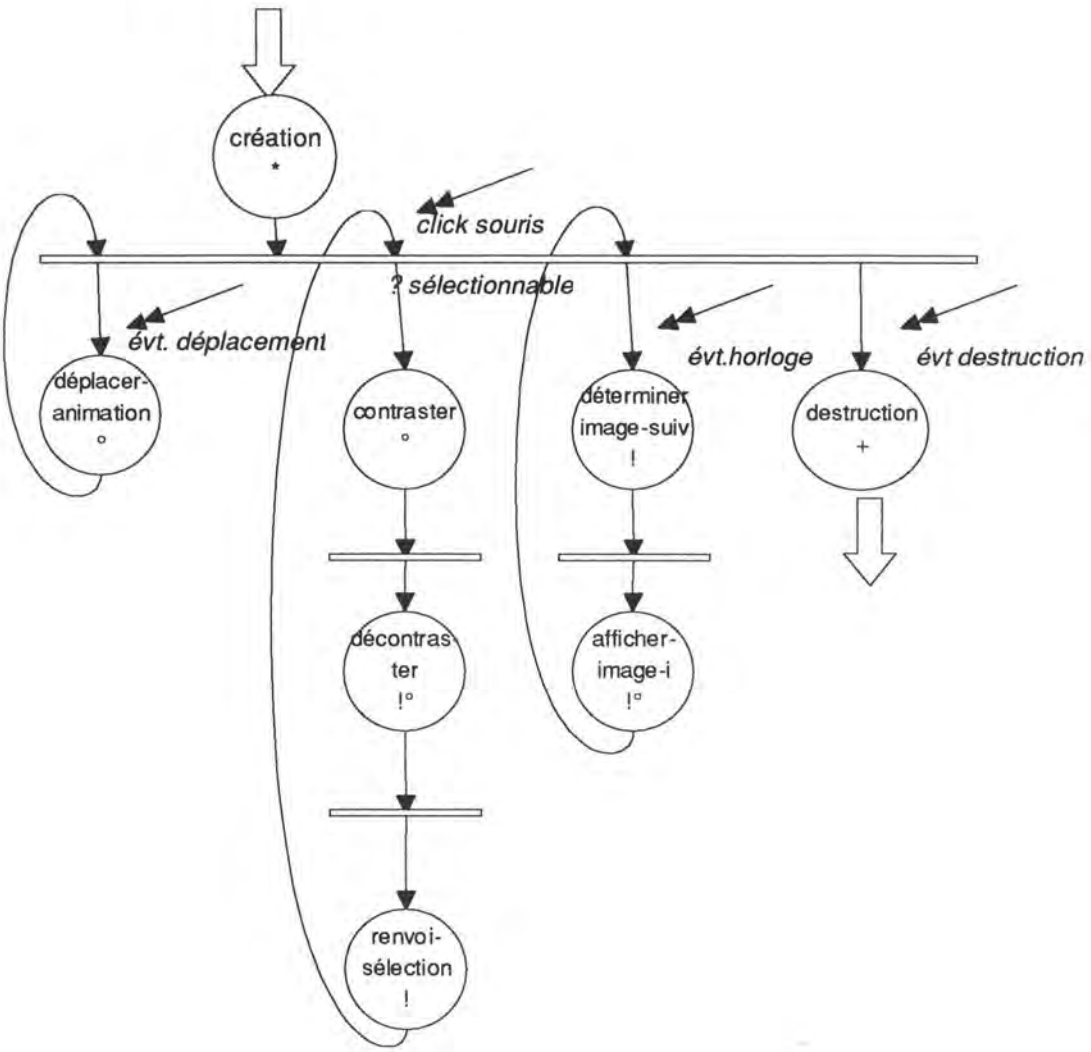


Diagramme de comportement:



3.2.Le son

Définition: Cet objet permet la manipulation de sons digitalisés. Il donne la possibilité à l'utilisateur de jouer des fichiers au format WAV. Etant donné l'existence d'un utilitaire livré avec l'environnement Windows gérant l'enregistrement de nouveaux sons, cette fonctionnalité n'a pas été intégré dans notre outil.

Diagramme matriciel:

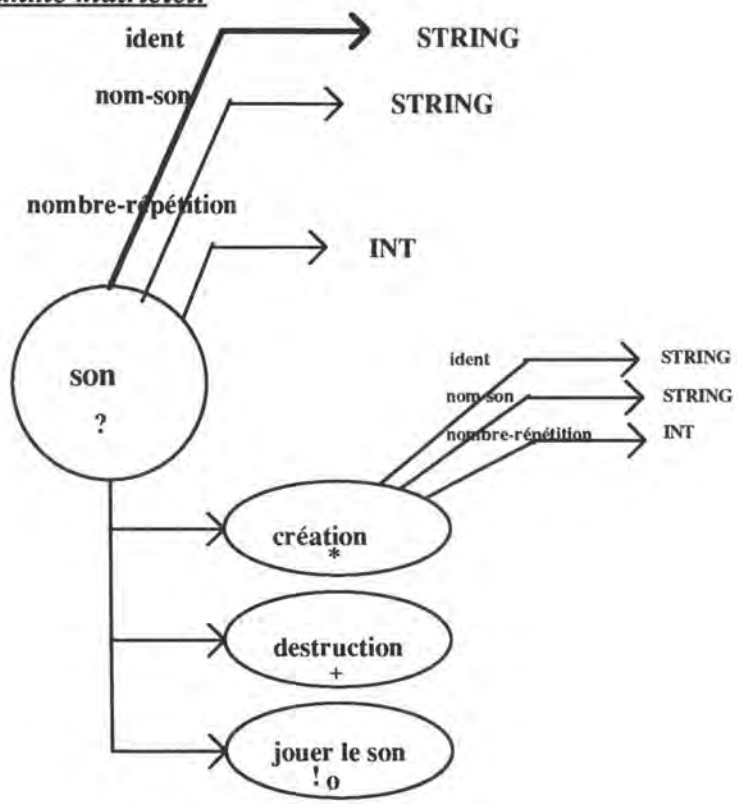
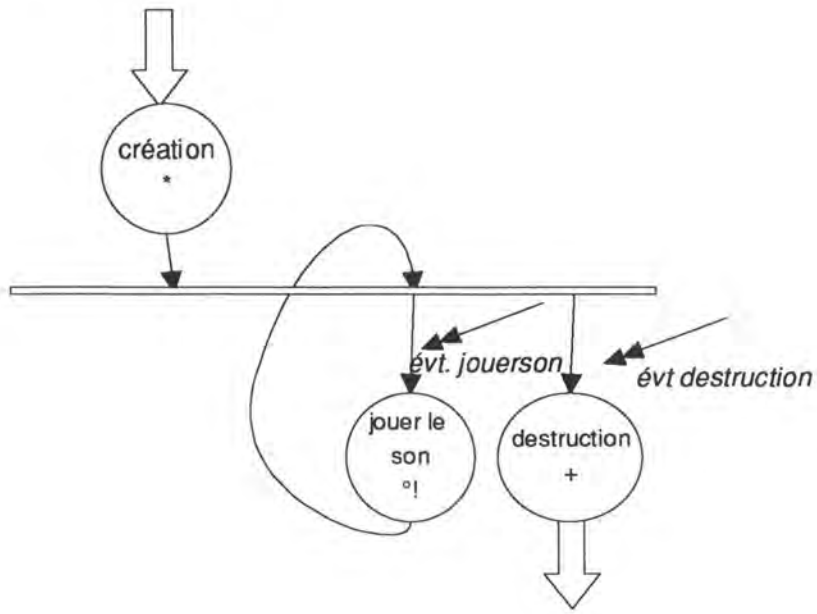


Diagramme de comportement:



3.3.Le texte

Définition: Le texte est un objet statique qui affiche à l'écran un libellé dans une police de caractère choisie par l'utilisateur. Ce libellé peut changer de valeur pendant la durée de vie de l'objet.

Exemple:

Cet outil, utilisé par exemple comme commentaire dans un écran proposant plusieurs types d'exercices, peut aider le moniteur dirigeant la session avec les enfants.

Une autre utilisation de cet outil est l'affichage du contenu d'une variable qui lui serait associée. Ainsi, l'utilisateur qui aurait par exemple entré son nom à la demande de l'ordinateur pourrait voir apparaître un message personnalisé de bienvenue reprenant son nom.

Diagramme matriciel:

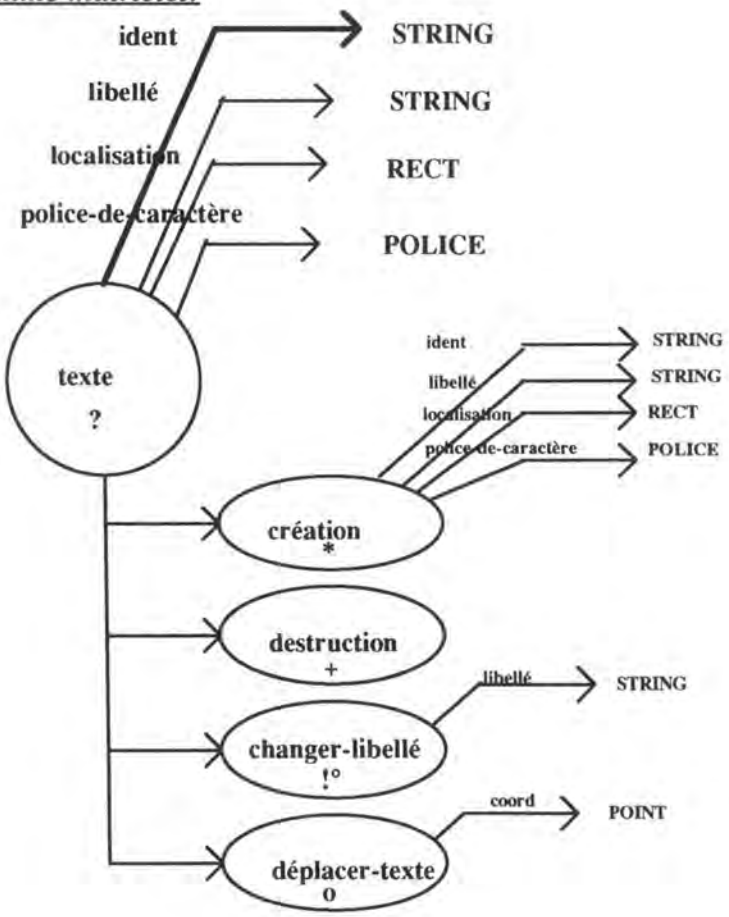
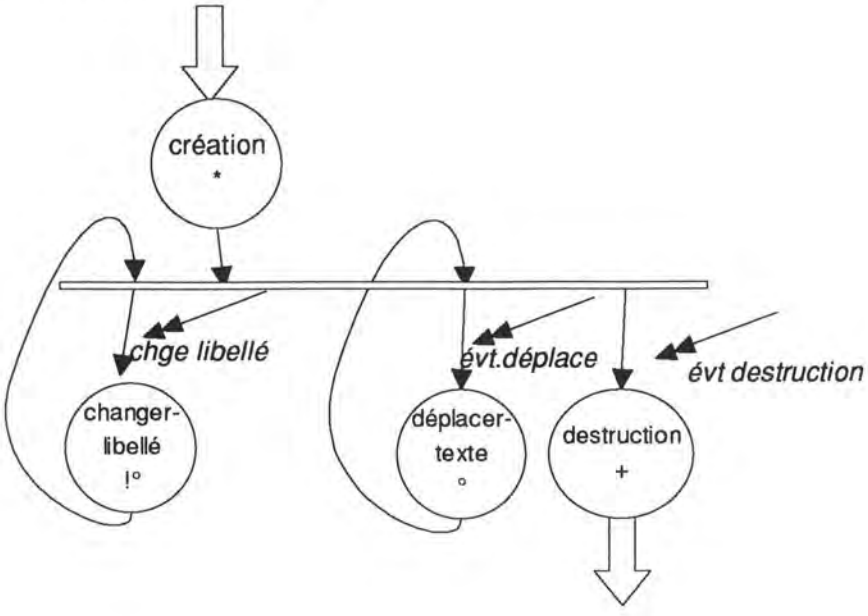


Diagramme de comportement:



Chapitre 5

L'environnement hardware et software

Ce chapitre est consacré aux différentes composantes de l'environnement dans lequel le logiciel auteur a été développé. Il expliquera les décisions prises dans la sélection du matériel ainsi que dans les choix des logiciels.

1.Le matériel utilisé

La machine retenue pour ce projet est un ordinateur de la gamme PC compatible IBM. Ce choix remonte à l'année passée. En effet, la boîte à outils a été implémentée sous Windows. Cet environnement est devenu en quelques années l'interface de référence de plus de 10 millions d'utilisateurs.
Avec l'introduction du son et des animations, notre boîte à outils devient Multimédia.
"Quel microprocesseur choisir ?", "Combien de mémoire vive faut-il ?", "Quelle est la taille minimale du disque dur ?", "Quelle carte sonore convient ?", sont autant de questions qui peuvent alors poser des problèmes et méritent donc d'être soulevées.
Pour simplifier la vie des utilisateurs souhaitant acquérir du matériel multimédia, Microsoft a lancé une norme: le MPC. Associé à cette norme, il existe un logo qui permet son identification (cfr. figure 1).

Figure 1: Le logo MPC.

Les premières spécifications du standard prévoyaient la configuration minimale requise pour adhérer au MPC. En voici les principales caractéristiques:

- un ordinateur à processeur 286 tournant à 12MHz,
- 2M en mémoire centrale,
- une carte VGA,
- une carte sonore,
- 30M de disque dur,
- une souris,
- ...

Ces spécifications ont été revues à la hausse après décembre 1992. En particulier, le processeur a été remplacé par un 386sx cadencé à 16MHz.

Cela conviendra pour nos besoins. Toutefois, il est à remarquer que pour des applications plus sophistiquées, une telle configuration sera totalement insuffisante.

Examinons maintenant les programmes qui ont permis l'élaboration du logiciel auteur: Windows 3.1. et Borland C++ 3.1.

2.Windows

2.1.Historique

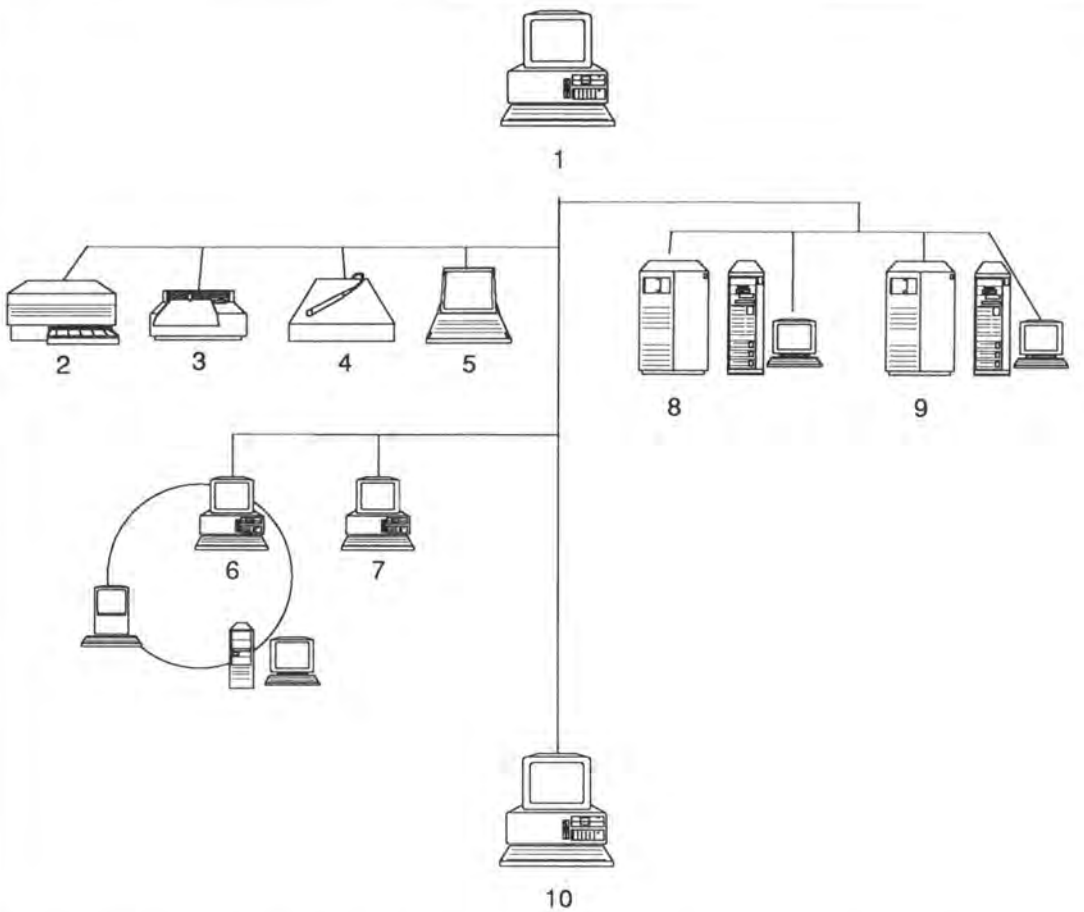
Dans les années '70, il apparaît clairement que l'interface entre l'homme et la machine doit être adaptée aux facultés cognitives et perceptives de l'utilisateur. De cette conviction à la création de la souris, des icônes et des interfaces graphiques , il ne faudra attendre que quelques années.

Vers 1985, ces idées vont être popularisées par l'arrivée du Macintosh sur le marché. Ce produit de la firme Apple est, à cette époque, destiné à un public un tantinet aisé.

Dans le monde du PC, on se dit qu'il serait bien d'avoir le même genre d'interface. De cette considération à la création de Windows, il n'y a qu'un pas et la première version sort en Novembre 1985. Toutefois, il faudra attendre la version 3 disponible en 1990 pour profiter pleinement du produit. L'évolution est loin d'être terminée puisque d'autres versions telles que la NT (New Technology) sont, à l'heure actuelle, sorties en version Béta[1].

Comme le montre la figure 2, Microsoft essaie de couvrir presque toutes les plate-formes.

[1] *version non définitive lancée sur le marché afin d'en détecter les 'bugs".*



1 Windows 3.1 initialement destiné aux PC's 386/486, aux portables, aux Pentiums, offre une large gamme de programmes.
2 Une version modulaire de Windows que l'on peut trouver uniquement sur des systèmes TANDY.
3 WinPad, qui a déjà été annoncé, est destiné aux PC's sans clavier.
4 Windows for Pens ajoute la reconnaissance de l'écriture manuscrite à la plupart des applications.
5 Mobile Windows, annoncé lui aussi, apporte des fonctionnalités destinées spécialement aux ordinateurs portables.
6 Windows for Workgroups est l'environnement Microsoft pouvant gérer un réseau.
7 Windows 32s est un environnement performant utilisant un modèle de mémoire 32 bits. Il permet d'exécuter des programmes de la version NT.
8 Windows NT for CISC est destiné aux stations de travail. C'est un système d'exploitation très stable. Il est capable de faire du multiprocessing symétrique et supporte des applications DOS et Windows.
9 Windows NT for RISC est prévu pour tourner sur lastation DEC Alpha et sur la série Mips R4000. Il ne pourra pas faire tourner directement des applications DOS et Windows. Un émulateur sera toutefois disponible par la suite.
10 Chicago (aussi appelé Windows 4) est le successeur de Windows 3.1. Il n'a pas encore été annoncé. Il pourra gérer les threads, les sémaphores, les courbes de Bézier et demandera seulement 4MB de RAM

Figure 2: L'arbre "généalogique" de Windows.

Source: *" Windows, Windows Everywhere? ", in BYTE., Juin 1993, pp 72-94.*

2.2.La "philosophie" suivie par Windows

Windows est un système d'exploitation qui permet, même s'il a encore besoin du MS-DOS pour fonctionner, une manipulation simplifiée des traitements grâce au

mode graphique et à l'utilisation de la souris . Il constitue une couche hermétique au-dessus du MS-DOS.

Une grande particularité de Windows est sans aucun doute son rôle d'intégrateur. Il assure, d'une part, une présentation standardisée des applications.

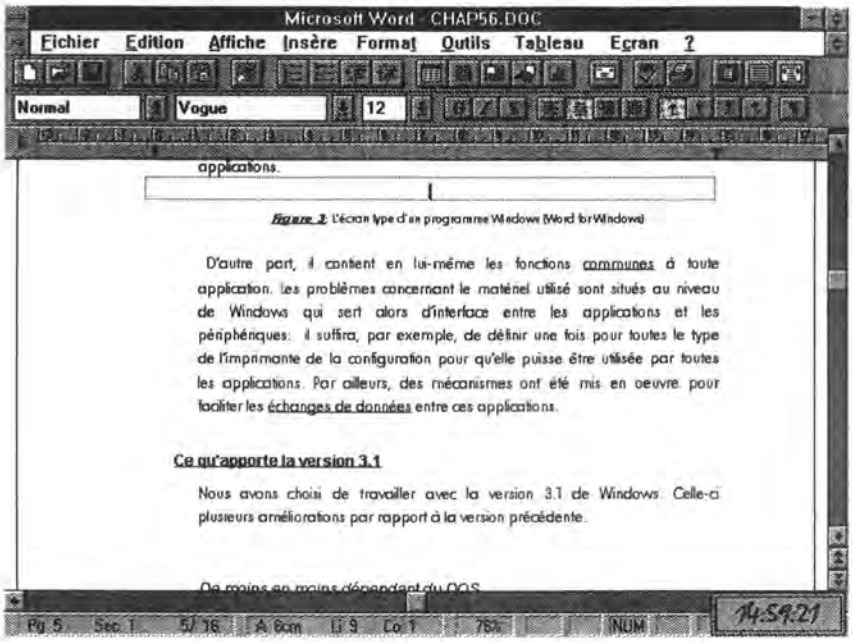


Figure 3: L'écran type d'un programme Windows (Word for Windows)

D'autre part, il contient en lui-même les fonctions communes à toute application. Ces fonctions sont réparties dans plusieurs modules qui forment ce qu'on appelle l'API de Windows. API étant l'abréviation pour Application Programming Interface.

Les problèmes concernant le matériel utilisé sont situés au niveau de Windows qui sert alors d'interface entre les applications et les périphériques: il suffira, par exemple, de définir une fois pour toutes le type de l'imprimante de la configuration pour qu'elle puisse être utilisée par toutes les applications. Par ailleurs, des mécanismes ont été mis en oeuvre pour faciliter les échanges de données entre ces applications.

2.3.Les différents modules de Windows

Windows n'est pas, contrairement à ce que l'on croit fréquemment, construit à partir de composants comme le gestionnaire de programmes, le panneau de configuration, ou le gestionnaire de tâches. Ceux-ci ne font pas plus partie de Windows que le COMMAND.COM ne fait partie du DOS.

Windows est construit au contraire à base de DLL (Dynamic Linked Librairies ou bibliothèques de liens dynamiques en Français). Trois d'entre elles constituent le coeur de l'API de Windows. Dans les premières versions, elles étaient confondues dans un même fichier mais depuis la version 3.0, ces modules sont séparés.

Il s'agit de:

- KERNEL: Il effectue les services du système, il s'occupe de la gestion de la mémoire, de la gestion des tâches et des liens dynamiques. Il existe trois versions de ce module. A savoir, KERNEL.EXE pour le mode réel (disparu depuis la version 3.1), KRNL286.EXE pour les machines 286 ou le mode standard, et KRNL386.EXE pour les processeurs 386 et plus.

- GDI ou Graphical Device Interface: Il s'occupe, comme son nom l'indique, de l'interface du périphérique graphique. Il fonctionne avec un pilote de périphérique DISPLAY comme VGA.DRV pour afficher du texte, des dessins, etc.

- USER: Il garantit les services de l'interface utilisateur. Cela comporte entre autres la création des fenêtres, l'envoi des messages, etc.

Dans Windows 3.1, un autre module important vient s'ajouter. Il s'agit de celui gérant les extensions multimédias. Toutes les fonctions s'y afférent sont regroupées dans la bibliothèque de liens dynamiques MMSYSTEM.DLL.

La figure 4 montre les relations entre les applications Windows, les modules principaux et les drivers de périphériques indispensables à la bonne marche du système.

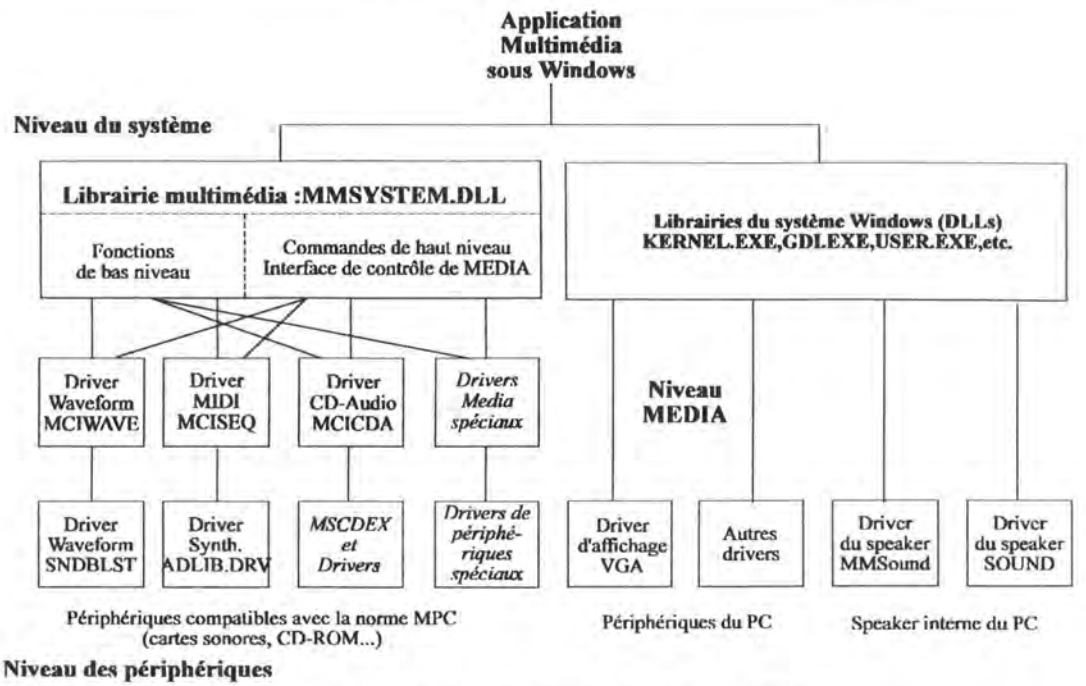


Figure 4: Les relations des Dlls et des drivers Windows.

Source: *Discover Windows 3.1. Multimedia, R. Jennings, 1992, Que Corporation ,Carmel,p141.*

2.4.Deux notions importantes de Windows: les fenêtres et les messages.

2.4.1.La notion de fenêtre

La fenêtre joue un rôle fondamental dans l'environnement Windows[2].

Une fenêtre n'est pas seulement un rectangle sur l'écran dans lequel une application effectue des affichages. Elle contient un certain nombre de caractéristiques et de fonctions traitant les messages en provenance de Windows. On retrouve donc ici une vue orientée objet qui est, par ailleurs, constamment présente dans Windows.

On peut relever comme caractéristiques principales:

- l'icône qui est affichée lorsque la fenêtre de l'application est réduite en icône,

[2]*De façon plus générale, la fenêtre joue un rôle important dans tous les environnements graphiques, qu'il s'agisse de Presentation Manager pour l'OS/2 ou encore de Mtof, OSF ou Open Look pour Unix ou l'un de ses dérivés.*

- le menu qui est affiché dans la partie supérieure de la fenêtre sous la barre de titre,
- les pinceaux (*brush* en anglais) pour colorier le fond de la fenêtre,
- les stylos (ou *pen* en anglais) pour dessiner ou écrire,
- les coordonnées sur l'écran.

Citons également quelques fonctions adressées à la fenêtre:

- au démarrage et lors de la fermeture de l'application,
- lors d'une modification de la taille de la fenêtre,
- à l'occasion d'un clic souris, d'une frappe d'une touche
- de façon plus générale, à l'occasion de divers événements.

2.4.2.La notion de message

Le mécanisme de communication (par messages) entre Windows et les programmes constitue sans aucun doute la différence essentielle (au point de vue de la programmation) entre une application traditionnelle et une application Windows.

Lorsqu'un programme traditionnel doit lire une réponse introduite au clavier, il demande au système d'exploitation: "Donnez moi le caractère qui a été ou va être frappé au clavier". C'est donc le programme qui appelle une fonction du système d'exploitation. Le programme garde ainsi le contrôle des opérations.

Un programme Windows ne fonctionne pas de cette manière. Il est informé par un message qu'un caractère vient d'être frappé ou que l'utilisateur vient de cliquer à tel emplacement de l'écran ou sur tel bouton de commande. Un programme écrit pour Windows devra réagir à ces messages. Dans ce cas, c'est l'utilisateur qui a le contrôle des opérations. Windows se charge de détecter les opérations effectuées par l'utilisateur, détermine à quel programme s'adresse l'opération (puisque nous sommes dans un environnement multitâche) et appelle alors une fonction du programme en question. Cet appel est accompagné d'une information appelée "message". Il s'agit en fait de l'argument de la fonction.

Contrairement aux programmes traditionnels, c'est bien Windows qui appelle une fonction du programme.

2.5.Ce qu'apporte la version 3.1

Nous avons choisi de travailler avec la version 3.1 de Windows. Celle-ci comporte plusieurs améliorations par rapport à la version précédente.

2.5.1.De moins en moins dépendant du DOS

Pour l'instant, Windows se base toujours sur MS-DOS pour fonctionner. Ces fonctions de très bas niveau sont appelées par le système des interruptions. Cela est bien sûr transparent pour l'utilisateur. Avec la version 3.1, Microsoft se détache peu à peu du DOS. Ainsi, la gestion de la mémoire, du disque dur de l'affichage, des périphériques se fait grâce à Windows.
L'objectif final de Microsoft est de lancer sur le marché un produit totalement indépendent du MS-DOS. Il serait alors un système d'exploitation à part entière dont les capacités promettent.

2.5.2.OLE (Object Linking and Embedding)

Cette technologie intégrée dans Windows 3.1. permet de changer ses habitudes de travail. Elle donne la possibilité de créer, dans un seul programme, des documents composites à partir de données provenant d'applications aux fonctions différentes. Grâce à cela, vous pouvez, par exemple, rédiger un rapport avec Word pour Windows II dans lequel vous y ajouterez un dessin créé dans Draw, un graphique provenant d'Excel, et pourquoi pas des annotations vocales numérisées avec l'accessoire Enregistreur de sons. L'astuce réside dans le fait que ces éléments sont attachés au programme qui a servi à les créer et leur mise à jour en est donc simplifiée. Microsoft, qui avait déjà lancé cette idée de liens dynamiques avec les DDE, propose maintenant les OLE. Regardons les différences entre ces deux méthodes:

- le concept d'***échange dynamique de données*** ou DDE en abrégé ("Dynamic Data Exchange") lie les objets de la manière suivante. Lorsqu'on modifie un

objet *grâce à son programme source*, le lien modifiera automatiquement l'objet importé. Il est à remarquer que cet objet importé appartient *toujours* à son programme d'origine.

- le concept de l'incorporation d'un objet appelé OLE ("Object Linking and Embedding") va quant à lui plus loin. Cette fois, l'objet importé appartient bien au programme "hôte" et fait partie intégrante du document. Il ne s'agit donc plus d'un simple lien mais d'une *incorporation*. En outre, l'utilisateur ne doit plus se préoccuper du programme source puisque l'objet lui-même s'en souvient. Pour l'éditer, il suffit de "double-cliquer". L'objet apparaît alors dans une fenêtre du programme source. De retour dans le logiciel "hôte", vous obtiendrez l'objet modifié.

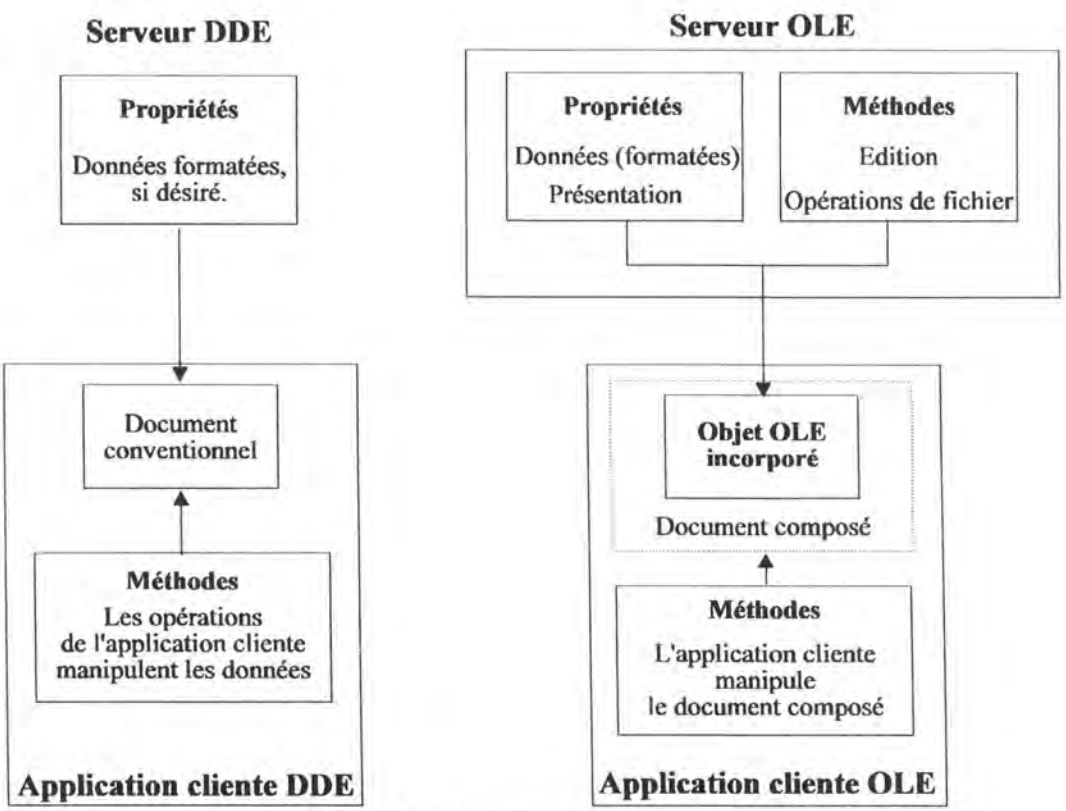


Figure 5: Les différences entre DDE et OLE.

Source: *Discover Windows 3.1. Multimedia, R. Jennings, 1992, Que Corporation ,Carmel,p220.*

Une petite évolution du standard OLE est prévue pour les prochaines versions de Windows. Le programme source apparaitra non plus dans une fenêtre séparée mais bien dans le programme "hôte": le menu du programme

source remplacera, par exemple, le menu du programme "hôte" pendant la durée de la mise-à-jour de l'objet.

2.5.3.La technologie True Type

Un problème rencontré lors de l'affichage et l'impression des caractères est sans aucun doute l'effet d'escaliers dû aux changements d'échelle. Pour solutionner ce problème, les concepteurs de Windows ont introduit dans la version 3.1 la gestion de polices vectorielles.

Figure 6: Les différentes polices de caractères vectorielles True Type.

Le résultat se traduit par une meilleure lisibilité des documents et le respect de la conformité entre l'affichage et l'impression, quels que soient le type de moniteur et de périphérique d'impression utilisés.

De plus, True Type apporte la compatibilité entre les documents Windows et les documents Macintosh, puisque cette technologie est également disponible sur les ordinateurs d'Apple.

Intégrées au système, les polices de caractères vectorielles True Type sont accessibles dans toutes les applications Windows. Cinq familles de polices sont livrées d'origine et il est possible d'en ajouter grâce à l'option "Police" du "Panneau de configuration" de Windows.

2.5.4.Plus fiable et plus rapide

La version 3.0 de Windows comportait encore quelques erreurs de fonctionnement qui bloquaient tout le système. Pour éviter cela, la version 3.1 intègre un dispositif de contrôle qui vérifie la validité des messages envoyés par les applications. Si ces messages menacent de provoquer une erreur, Windows empêche leur exécution. Il affiche une boîte de dialogue qui indique le type de problème rencontré et l'application qui en est responsable. En général, Windows propose alors de fermer l'application fautive afin que cela ne recommence plus. En plus de cela, il est possible d'interrompre une application qui se serait bloquée en tapant simultanément sur les touches Ctrl-Alt-Del.

La version 3.1. a également subi quelques optimisations. Une meilleure gestion de la mémoire, des performances accrues des opérations sur disque dur ainsi que des optimisations du programme même ont permis une plus grande vitesse d'exécution.

2.5.5.Le son

En plus des accessoires habituels fournis avec Windows, l'environnement de Microsoft comporte dans sa version 3.1 plusieurs extensions "multimédias" principalement destinées au traitement de données sonores. Toutes les requêtes pour obtenir des services multimédias passent par une des nouvelles librairies à liens dynamiques de la version 3.1.: MMSYSTEM.DLL.

Grâce à ces extensions multimédias, on peut, par exemple, continuer à travailler tout en écoutant de la musique, insérer des annotations vocales dans des documents et même préparer des présentations ou des animations accompagnées d'effets sonores ou d'une musique de fond.

Les sons numérisés peuvent être modifiés à l'aide du programme "Enregistreur de sons". Il se présente sous la forme d'un tableau de commande accompagné d'une fenêtre dans laquelle se dessine la forme d'onde du son (cfr. figure 7).

Figure 7: Un accessoire livré avec Windows 3.1: l'enregisteur de sons.

Plusieurs traitements sont possibles: augmenter ou réduire le volume, accélérer ou ralentir l'exécution, ajouter de l'écho, jouer le son à l'envers... Pour enregistrer un son , il faut disposer d'une carte sonore. Windows est livré avec des gestionnaires de cartes sonores mais il est possible de charger les programmes de pilotage fournis par le fabricant du périphérique.

Toutes ces caractéristiques ont bien entendu apporté leur poids dans la décision d'utiliser la version 3.1 de Windows.

3.Borland C++

3.1.Pourquoi le C ?

L'idée de base qui a dirigé la définition du langage C a été d'offrir un langage qui permette à la fois une programmation de haut niveau, comme en Pascal, et de bas niveau comme en Assembleur. La fusion de ses deux aspects, *a priori* peu compatibles, a sans aucun doute été un des facteurs essentiels de réussite pour ce langage.

Ces caractéristiques essentielles du C permettent d'une part la ***portabilité*** , c'est-à-dire la possibilité d'utiliser le même texte de programme pour réaliser une même application sur des machines ou des environnements différents, et d'autre part l'***efficacité*** qui est la faculté d'exploiter au mieux les ressources offertes par la ou les machines sur lesquelles le logiciel sera implanté.

Une autre caractéristique qui doit retenir notre attention est la ***réutilisabilité***. Il s'agit de pouvoir réutiliser des développements réalisés pour une application dans d'autres contextes. Cela nécessite un effort supplémentaire lors de la conception et de l'implémentation afin de garantir une certaine généralité aux modules développés.

Les choix qui ont amené les étudiants de l'année passée à utiliser le langage C ont été expliqués dans leur mémoire (cfr [DEMO,92]). Ils illustrent bien ces notions que nous venons de voir :

- les objets définis dans la boîte à outils nécessitaient une programmation de bas niveau (efficacité);
- il fallait un outil capable d'élaborer des bibliothèques (réutilisabilité).

Par ailleurs, il fallait tenir compte du facteur temps. La documentation concernant la programmation Windows est souvent dédiée au C. Une transition dans un autre langage aurait causé des pertes de temps intolérables.

Les outils de l'an passé ont donc été écrits en C.

3.2.Pourquoi le C++ ?

Le maniement des outils dans quelques programmes a permis de déceler un problème typique de Windows.

Celui-ci utilise la notion de **ressources**. Les ressources sont les composantes d'un programme qui sont créées en dehors du programme lui-même. Il peut s'agir d' icônes[3], de curseurs ou encore d'images bitmaps.

A chaque exécution des exercices utilisant la boîte à outils, le nombre de ressources libres du système diminuait. On a pu s'en rendre compte facilement grâce à la boîte de dialogue "A propos de..." du Gestionnaire de programmes (cfr. figure 8).

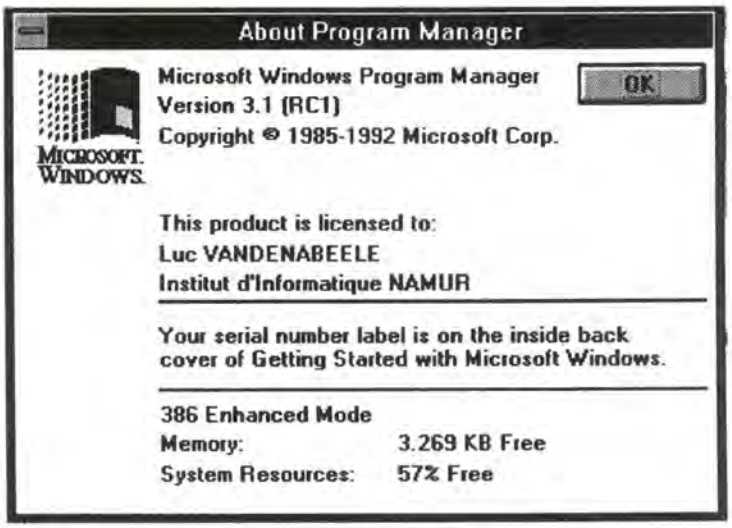


Figure 8: La boîte de dialogue "A propos de.." du Gestionnaire de programmes indique le pourcentage de ressources libres du système.

Pour utiliser ces ressources, le programmeur doit en faire la "demande" à Windows. Ainsi par exemple, pour afficher une image à l'écran, on doit réserver un "contexte de périphérique" lié à l'unité d'affichage de l'ordinateur. Il est important de libérer ce "contexte de périphérique" dès que possible. Lorsqu'on oublie de le faire, cela peut causer à terme un blocage brusque du système.

[3] *au sens de Windows. A ne pas confondre avec l'icône de notre boîte à outils...*

Après avoir essayé, en vain, de repérer d'où pouvaient provenir ces "fuites" de ressources disponibles du système, la réécriture des outils en C++ semblait nécessaire. Ceci pour les raisons suivantes:

- Les concepts de **constructeurs** et de **destructeurs** liés à la création et à la destruction des instances de classes permettent d'effectuer des opérations spécifiques concernant par exemple l'état de la mémoire. En C++, le destructeur, qui est une fonction membre, a une particularité intéressante: il est appelé *automatiquement* lors de la suppression de l'instance. Les risques d' "oublis" de remise en état initial du système sont donc diminués.

- La modélisation objet est adéquate dans notre cas. Que se soit au niveau des outils ainsi qu'au niveau de l'application elle-même. Tous les mécanismes liés à la programmation orientée objet s'avèrent, ici d'une grande utilité. Le fait qu'un langage orienté objet n'ait pas été utilisé l'an passé pour l'implémentation des outils peut s'expliquer par le manque de temps. L'apprentissage d'un tel langage représentait, en effet, un investissement que les étudiant précédents ne pouvaient pas se permettre.

- Finalement, - et cette raison n'est certainement pas à négliger - , la compatibilité qui existe entre le C et le C++ permet une transposition minimale de ce qui a été écrit l'an passé.

4.Conclusion

Dans ce chapitre, nous avons exposé les raisons qui ont dirigé le choix de l'environnement matériel et logiciel.

La configuration matérielle minimale sur laquelle tournera le logiciel devra suivre la norme MPC.

Le logiciel auteur sera destiné à l'environnement Windows 3.1 et sera développé sous Borland C++ 3.1.

Chapitre 6

L'implémentation

Le but de ce chapitre est de montrer quelques caractéristiques de l'implémentation. Il sera consacré aux problèmes de maintenance , à la hiérarchie des classes du système, ainsi qu'aux problèmes plus techniques liés à la programmation sous Windows.

1.La maintenance

Le projet qui, je le rappelle, a débuté voici deux ans, a nécessité une maintenance des outils de base. C'est pourquoi j'ai introduit quelques notes sur ce sujet qui mérite notre attention en tant qu'informaticiens.

L'importance de la maintenance dans le cycle de vie d'un système a souvent été soulignée dans la littérature:
"... Canning (1972) remarque que "la plupart des utilisateurs d'ordinateurs reconnaissent qu'environ 50% des dépenses en matière de programmation partent dans la maintenance d'applications déjà en service". Cela devient un problème de management à cause des impacts sur les budgets et sur les bénéfices. Ditri, et al (1971) cite une figure similaire. [...] Riggs (1969) estime, quant à lui, que la demande de maintenance sur les systèmes et les ressources de programmation varient de 40 à 60% pour la plupart des compagnies qui ont des systèmes informatiques depuis plusieurs années..." ***Source**: Software Maintenance Management, Lientz et al, 1980, pp4-9.*

Que se cache-t-il vraiment derrière ce terme de 'maintenance' d'un logiciel ? En général, la littérature répond à cette question en distinguant deux types d'activités:

- Le premier type d'activités concerne les ***modifications*** à apporter au système. Celles-ci peuvent survenir lorsque les spécifications du système informatique ont changé, reflétant par exemple des modifications du monde extérieur.

- Le second type d'activités s'occupe quant à lui des ***corrections*** des erreurs qui auraient pu se glisser dans le système lors des implémentations précédentes.

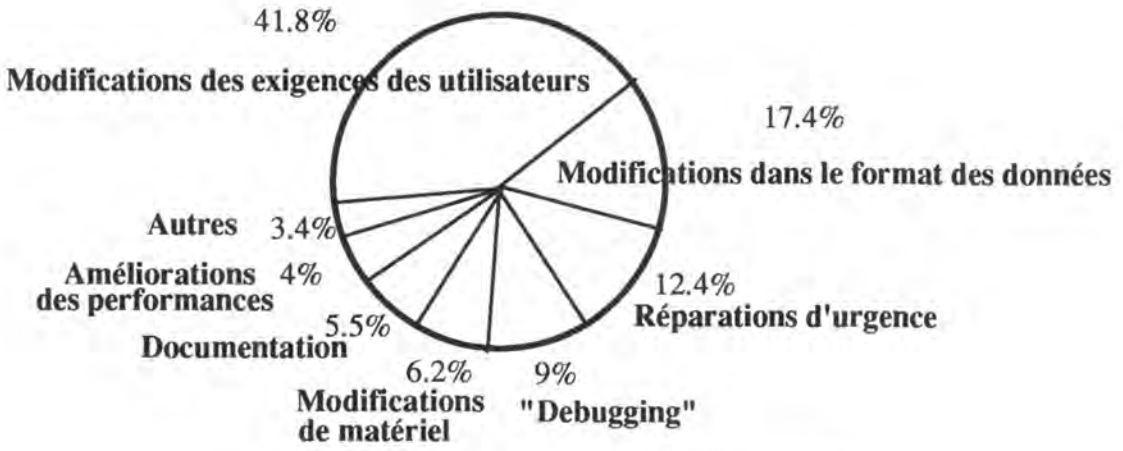


Figure 1: La répartition des coûts de maintenance.

Source:Object-oriented Software Construction, Meyer,1988, Prentice Hall International, p.8.

La figure 1 montre la diversité des domaines couverts par la maintenance de logiciels. Ces chiffres, basés sur une enquête englobant plus de 480 développements de logiciels , indiquent que plus des 2/5 des activités de maintenance concernent les extensions et les modifications réclamées par les utilisateurs. Cette partie est pratiquement inévitable.

Le deuxième élément par ordre d'importance concerne les effets dus aux modifications des formats des données. De tels changements peuvent entraîner de coûteuses adaptations dans les programmes. Un exemple connu fut, il y a quelques années, l'introduction par l'Office des Postes des USA d'un code postal comportant 9 chiffres alors que le format standard en utilisait 5. Presque tous les programmes manipulant des adresses durent être réécrits à grands coûts. Une solution à ce problème est donnée par ce qu'on appelle les types de données abstraites: l'accès aux données ne se fait plus directement par une implémentation physique mais par l'appel de propriétés externes.

Un autre élément significatif est relatif aux problèmes de documentation. Il paraît évident qu'un code source bien documenté peut se révéler comme un atout pour la maintenance de celui-ci.

Les autres éléments peuvent également indiquer certaines choses. Par exemple, le faible taux des améliorations de performances laisse supposer que les concepteurs ne désirent plus modifier le système dès celui-ci fonctionne correctement.

Dans le projet qui nous concerne, on retrouve bien les deux types d'activités de la maintenance.

D'une part, plusieurs caractéristiques des objets de la boîte à outils ont été ***modifiées*** suite aux demandes des logopèdes du Trèfle.

Ainsi, par exemple, dans la première version, les pictogrammes représentant graphiquement l'outil icône devaient être impérativement connus au moment de la compilation du programme (puisque ceux-ci étaient inclus dans le fichier des ressources du projet). Cette restriction était intolérable dans le cadre d'un logiciel auteur. Actuellement, les pictogrammes peuvent être choisis lors de l'exécution du programme et sont chargés en mémoire à partir de fichiers présents sur le disque dur. Une autre modification concerne la taille de ces pictogrammes. L'outil icône peut maintenant être redimensionné à volonté lors de l'exécution du programme. Cette possibilité permet une meilleure adaptation aux capacités visuelles de l'utilisateur.

Des ***corrections*** durent également être apportées aux outils. Je citerai simplement deux exemples qui ont pu être détectés lors des séances de tests avec les logopèdes du Trèfle.

Premier exemple

Symptôme: Lorsque l'utilisateur quitte un exercice utilisant la boîte à outil et revient à Windows, le système est instable et se bloque après un bref délai.

Explication: A la fin de l'exécution d'une application, le système doit être remis dans l'état initial. Cela se fait grâce à l'appel d'une fonction de l'API WINDOWS: PostQuitMessage. Cette fonction admet un argument qui est ce qu'on appelle la valeur de retour du programme. Si on ne termine pas l'exécution de l'application par l'appel à cette fonction, les conséquences sont pratiquement immédiates.

Cette première correction fut rapidement apportée.

Deuxième exemple

Symptôme: Le pourcentage de ressources disponibles du système diminue après chaque exécution d'une application utilisant la boîte à outils. A terme, le système se bloque puisque plus aucune ressource n'est disponible.

Explication: Le pourcentage de ressources système disponible est un chiffre très important pour l'utilisateur de Windows, car il détermine - bien mieux que le total de mémoire disponible - le nombre d'applications qui peuvent tourner au même moment.
J'ai déjà signalé que le gestionnaire de fenêtres (USER) et l'interface des périphériques graphiques (GDI) constituaient deux modules principaux de Windows (cfr chapitre 5). Ils sont donc utilisés par tous les autres modules du système. Ainsi lorsque une fenêtre ou un menu est créé, c'est une allocation supplémentaire qui diminue la taille mémoire du tas local de USER. De même, chaque numéro d'accès (handle), pinceau, stylo, région, fonte de caractères, ou image matricielle (bitmap) apparaît comme une allocation dans le tas local de GDI.
Les conséquences sont considérables puisque, dans la version 3.0, toutes les applications doivent se partager un seul tas local de 64K pour USER et un seul de 64K pour GDI. Même avec des mégaoctets de mémoire disponibles en mode protégé, le nombre d'applications pouvant tourner simultanément (et leur stabilité) reste subordonné à ces deux barrières de 64K. Dans la version 3.1, plusieurs tas locaux ont été attribués à ces deux modules, résolvant ainsi une partie du problème. Malgré cela, une limite supérieure subsiste pour le nombre de fenêtres, de menus, de stylos, de pinceaux, etc. pouvant être créés à un instant donné.

De ce fait, il est impératif de libérer les ressources dès qu'elles n'ont plus leur utilité dans l'application.

Apparemment, c'est un oubli de ce genre dans la boîte à outils qui a causé ce que j'appellerai une "fuite" des ressources disponibles. Après avoir passé un temps non négligeable à essayer de repérer dans le code initial l'origine de cette "fuite", j'ai pris la décision de transcrire systématiquement les outils en C++. Grâce, entre autre, aux mécanismes de **constructeur** et **destructeur** des classes C++, l'allocation et la libération des ressources pouvaient se faire automatiquement. Ces mécanismes réduisaient donc au maximum les risques d'oublis.

Dans la version actuelle, le pourcentage de ressources disponibles ne diminue pas d'une exécution à l'autre des exercices utilisant la boîte à outils écrite en C++.

2.L'architecture des classes

La théorie qui se rapporte aux mécanismes orientés objets a été développée dans le travail de l'année passée (cfr.[DEMO,92], chapitre 3). Nous n'y reviendrons donc pas. L'objectif de cette section est de présenter brièvement l'architecture objet du logiciel.

Cette architecture se base sur une librairie (DLL) fournie avec le Borland C++: l'*Object Windows Librairy* (ou de l'*OWL* en abrégé) Celle-ci facilite l'utilisation des objets internes à Windows. De nombreuses classes du logiciel auteur héritent des classes de l'*OWL*.

La figure 2 reprend de manière schématique et simplifiée la hiérarchie des classes du logiciel auteur.

D'après cette figure, on remarque que pratiquement la totalité des classes dispose d'un ancêtre commun: l'*Object*. Celui-ci est une classe abstraite qui donnent les bases de toute la hiérarchie des objets Windows. Un exemple de méthode liée à cette classe permet la comparaison de deux instances d'objet.

La classe *TWindowsObject* définit le comportement fondamental de l'interface des objets de type fenêtre. Un des rôles de cette classe est de piloter les messages destinés aux divers éléments de la fenêtre. Les deux descendants directs de cette classe sont d'une part *TDialog* (qui permet la gestion des boîtes de dialogue) et *TWindow* (qui donne le comportement spécifique d'une fenêtre). C'est de cette dernière classe que vont être dérivées les éléments constitutifs de l'interface du logiciel auteur: *TRibbon*, *StatusBar*, *TAuteurWin* et *Element* (qui est la classe abstraite ancêtre des objets de la boîte à outils).

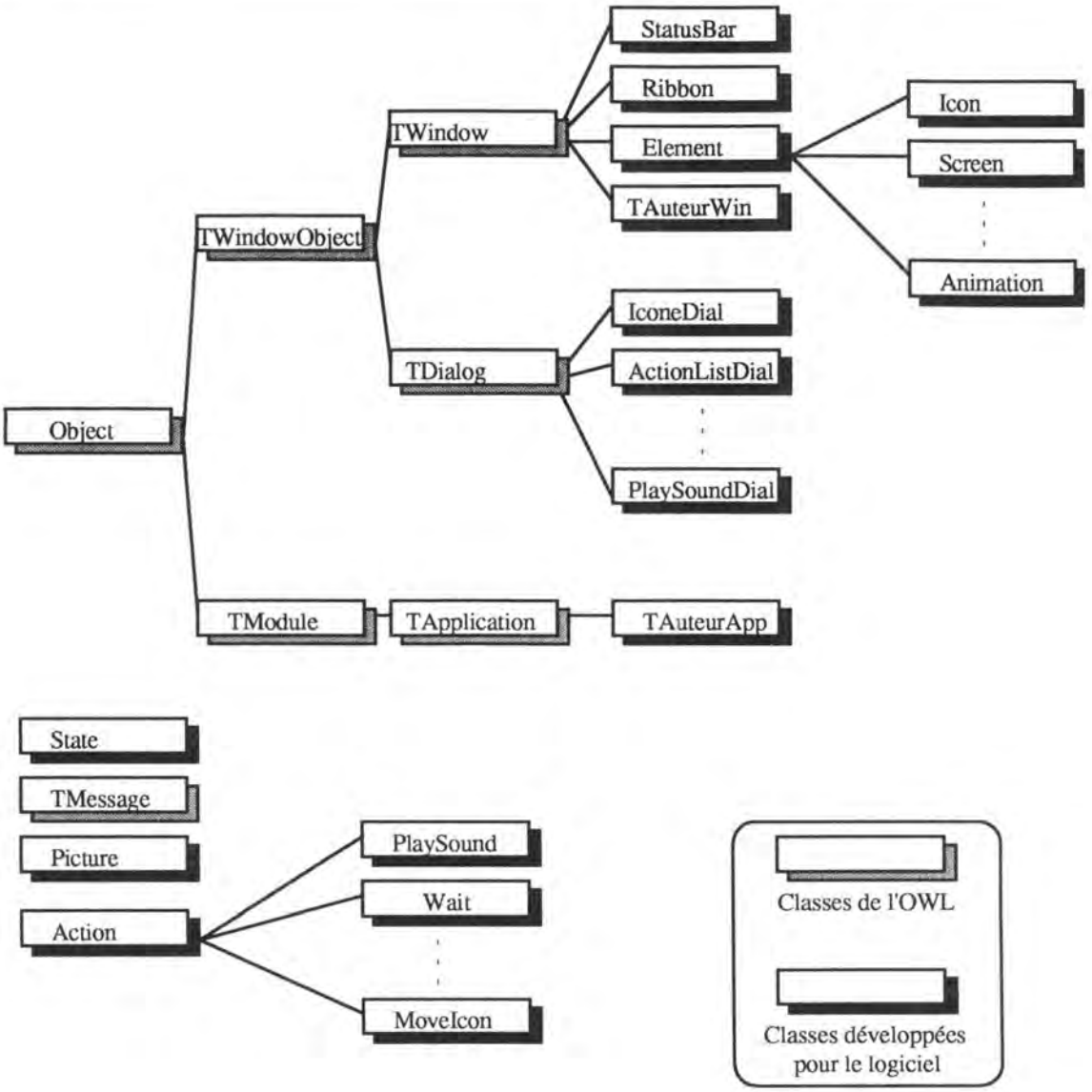


Figure 2: La hiérarchie (simplifiée) des classes du logiciel auteur.

Les classes *TModule*, *TApplication* et *TAuteurApp* déterminent le comportement de l'application. Elles gèrent entre autre les initialisations des instances des classes et le mécanisme des messages.

La classe TMessage contient les informations véhiculées à travers le système: autrement dit, les messages.

State donne certaines caractéristiques générales du système. C'est grâce à cette classe par exemple que l'on bascule du mode conception au mode exécution et réciproquement. C'est également elle qui renseigne les descendants de la classe *Elément* de l'état courant: cliquer sur un objet lorsque l'on est dans le mode conception aura pour conséquence l'apparition de la boîte de dialogue permettant la paramétrisation de l'objet alors que la

même opération effectuée dans le mode exécution déclenchera la liste des actions associées à l'objet en question.

La classe *Picture* a été implémentée pour rendre transparent la gestion des pictogrammes provenant de fichiers (cfr. section suivante). Ses méthodes permettent le chargement en mémoire et le redimensionnement des pictogrammes.

La dernière série de classes modélise les actions qui sont déclenchées lorsqu'un objet composant un exercice a été sélectionné. Cette série est composée d'une classe abtraite ancêtre (*Action*) et d'une classe descendante pour chaque action disponible.

3.Description de quelques problèmes rencontrés lors de l'implémentation

3.1.Les pictogrammes provenant d'un fichier

L'utilisation de pictogrammes provenant de fichiers a été une des premières modifications apportées à la boîte à outils. Le choix s'est porté sur les fichiers portant l'extension BMP (le standard en matière de bitmaps Windows). Le principal problème consistait à trouver la description exacte du format BMP.

Nous nous sommes basés sur *Programmation Windows en Turbo C++ et Borland C++, G. Leblanc, 1992, Eyrolles, p392.*

Un fichier BMP est composé de deux parties principales:

- La première partie est un en-tête dont le format correspond à une structure de donnée fixe. Les renseignement que l'on trouve dans cet en-tête concernent la taille du fichier ainsi que le déplacement (à partir du début de cette structure) du début de la partie DIB du fichier.

- La seconde partie, que l'on appelle DIB (Device Independant Bitmap), est quant à elle composée de deux éléments:

- Un en-tête reprenant des renseignements tels que la taille du pictogramme, le nombre de bits nécessaires pour coder une couleur, le nombre de couleurs dans la table des couleurs, la table des couleurs, etc.

- La description des pixels formant le pictogramme.

La particularité de ces fichiers est la présence de la table des couleurs. Le pilote de la carte vidéo peut ainsi tenir compte des couleurs spécifiées dans le fichier et de celles qu'il est capable de générer. La valeur correspondant à un pixel représente un indice dans la table de couleurs.

Grâce à cette description, l'algorithme permettant le chargement d'un pictogramme en mémoire a pu être élaboré (cfr. figure 3).

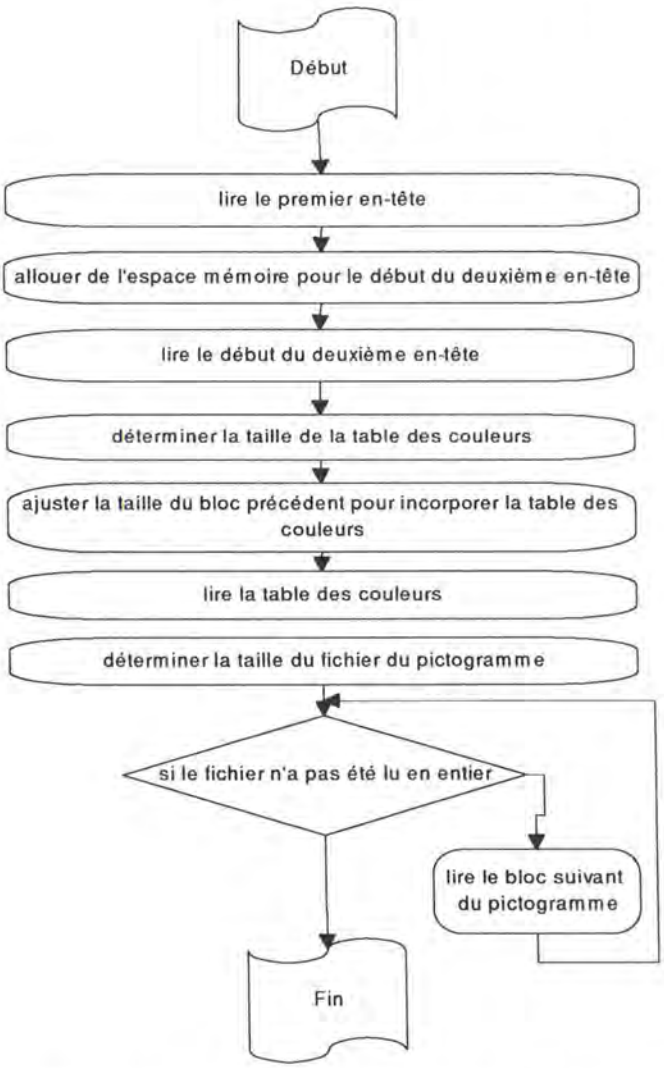


Figure 3: Algorithme réalisant le chargement en mémoire d'un fichier au format BMP.

3.2.Les problèmes de contrôle du temps

Certaines actions telles que le déplacement d'icône sur l'écran, l'animation graphique et le défilement automatique (lié au moyen d'interaction par interrupteurs) demandent un mécanisme contrôlant le temps.

Une fonction de l'API Windows en donne la possibilité. Il s'agit de *SetTimer*. Cette fonction provoque soit l'émission d'un message WM_TIMER soit l'appel d'une fonction à ***intervalles réguliers***.

Regardons maintenant l'utilisation de ce principe dans les mécanismes de déplacement d'icône, d'animations et de défilement automatique.

3.2.1.Le déplacement d'icône

Le déplacement d'un objet d'un point A vers un point B peut être décomposé en plusieurs étapes (cfr. figure 4).

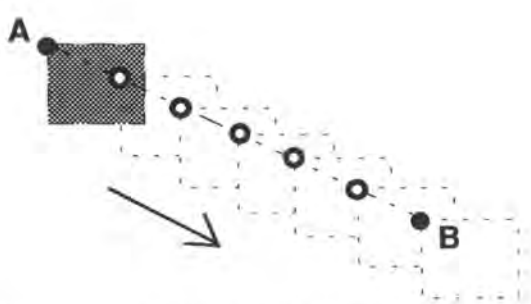


Figure 4: Trajectoire d'un objet se déplaçant du point A vers le point B.

Chaque étape consiste à calculer la nouvelle position de l'objet et à le placer à cette nouvelle position.

3.2.1.1.Le calcul de la nouvelle position

Etant données a1,a2 et b1,b2 , représentant respectivement les coordonnées des points A et B et N le nombre d'étapes désirées; les coordonnées z1 et z2 de la i^{eme} position (0<=i<=N) se calculent comme suit:

$$z1 = a1 + i * (b1-a1)/N$$

$$z2 = a2 + i * (b2-a2)/N$$

3.2.1.2.Le placement de l'objet à la nouvelle position

Le déplacement d'objet mérite d'être optimisé afin d'éviter le plus possible les phénomènes de saccades. Lors du déplacement, le fond caché dans la position précédente doit évidemment être restitué.

Considérons donc deux positions de l'objet en cours d'un déplacement élémentaire: R1 est la position précédente et R2 la nouvelle position.

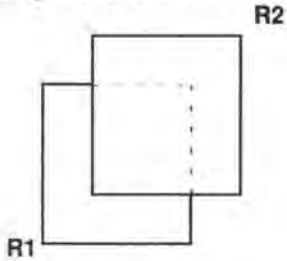


Figure 5: Un déplacement élémentaire.

Dans le rectangle R2, nous devons afficher le pictogramme associé à l'objet. Cette opération se fait grâce à la fonction de l'API Windows ***BitBlt***. Par contre, dans la partie de R1 qui n'est pas recouverte par R2, nous devons restituer l'image de fond. Pour cela, nous allons délimiter la zone sombre du schéma suivant:

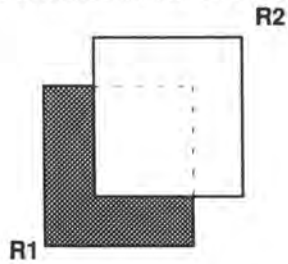


Figure 6: La zone à restituer.

Cette zone est obtenue par l'opération R1 "moins" R2. La fonction de l'API Windows correspondante s'appelle ***ExcludeClipRect***. En passant comme argument cette zone qui a la forme d'un L à la fonction chargée de dessiner le fond de la fenêtre, Windows limitera automatiquement l'affichage à cette zone.
Le déplacement de l'objet peut ainsi s'effectuer de manière satisfaisante, sans à-coup et sans fatigue pour l'oeil.

Lorsque l'objet a atteint la position de destination, Il suffit d'indiquer à Windows d'arrêter l'émission du message WM_TIMER pour stopper le déplacement.

3.2.2.Les animations

L'animation est constituée, selon le principe des dessins animés, de plusieurs pictogrammes défilant très rapidement à l'écran.

Chaque étape consistera donc à afficher le pictogramme suivant dans la liste. Lorsque le dernier pictogramme a été affiché, il convient d'indiquer à Windows de stopper l'émission du message WM_TIMER. Cela arrêtera l'animation.

3.2.3.Le défilement automatique

Le défilement automatique est un mécanisme intéressant pour les personnes éprouvant des difficultés à manipuler le clavier ou la souris.

Ce mécanisme consiste à déplacer de manière automatique le curseur indiquant la pré-sélection d'un objet. L'action de l'utilisateur est ainsi réduite au maximum. Lorsque le curseur arrive sur l'objet qu'il désire sélectionner, il lui suffit alors de valider la pré-sélection en appuyant sur un interrupteur.

La figure suivante, qui est reprise de l'exercice de coloriage (cfr. chapitre 2), montre les différentes étapes du défilement automatique.

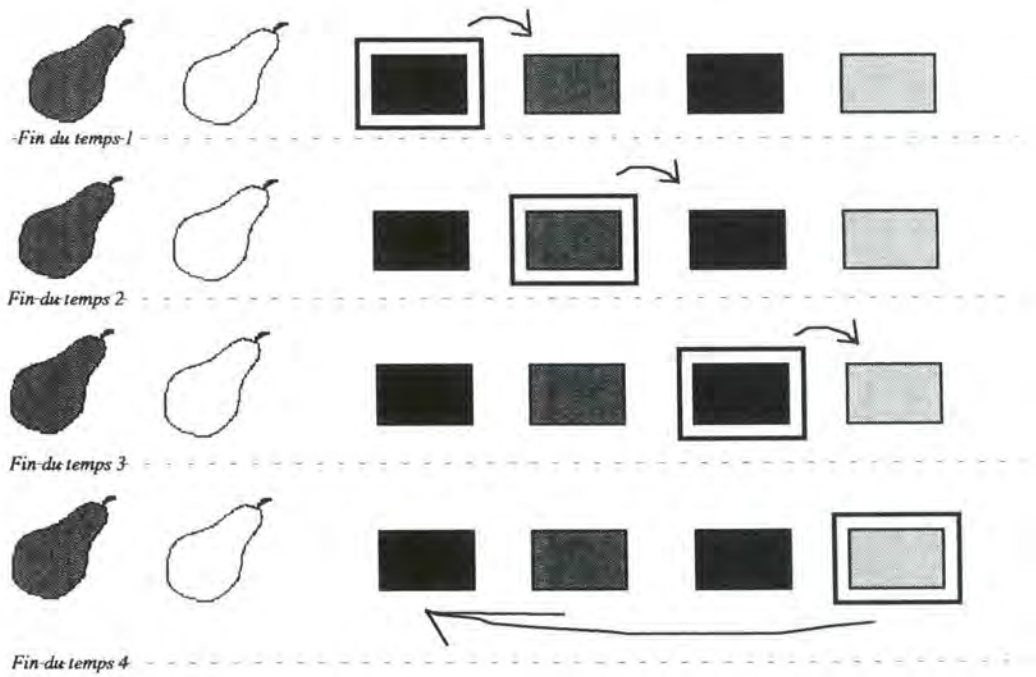


Figure 7: Le défilement automatique.

Chaque étape du défilement automatique consiste à rechercher dans la liste des objets affichés à l'écran le prochain qui possède la caractéristique *'sélectionnable'* et à déplacer le curseur vers cet objet.

Lorsque l'utilisateur confirme son choix en validant l'objet pré-sélectionné, la liste des actions associées à cet objet est exécutée. Ensuite, le défilement automatique peut reprendre.

3.3.Le mécanisme des "Hooks" pour les traces.

Pour implémenter les méthodes nécessaires à l'extraction des données décrivant les événements (clicks souris, touche appuyée etc...) qui se passent durant les sessions d'exercices, le mécanisme des "hooks" est idéal.

En fait, ce système donne non seulement la possibilité d'avertir une application des événements qui se produisent mais permet également d'agir comme filtre en empêchant certains événements.

Le principe est le suivant:
Lorsqu'un événement se produit, il se traduit automatiquement par l'émission d'un message. Si un "hook" correspondant au type du message envoyé est activé, celui-ci est intercepté. Une fonction associée au "hook" est alors appelée et traite le message. Le traitement fini, le message peut ou non être réintroduit dans le système.

Il existe 12 types de "hooks":

- ceux qui interceptent les messages envoyés à une fenêtre,
- ceux qui interceptent les messages reçus par une fenêtre,
- ceux qui indiquent qu'une touche du clavier a été enfoncée ou relâchée,
- ceux qui suivent les mouvements de la souris,
- ceux qui sont relatifs aux messages en provenance du matériel (à l'exception du clavier et de la souris).
- ceux qui filtrent les messages qui vont être traités par les menus,
- ceux qui filtrent les messages qui vont être traités dans l'application qui a installé le "hook",
- ceux qui enregistrent tous les messages présents dans la file d'attente de Windows,

- ceux qui se déclenchent à chaque fois qu'un événement est demandé par la file d'attente de Windows,
- ceux qui se déclenchent lorsqu'une manipulation de fenêtre a eu lieu (création, destruction, minimisation, maximisation, déplacement, etc),
- ceux qui indiquent des manipulations de l'application principale de Windows (le gestionnaire des programmes),
- ceux qui sont appelés avant n'importe quel autre "hook".

Pour plus de détails sur les types de "hooks", le lecteur intéressé pourra consulter Windows 3.1: A Developer's Guide, Richter,1992,M&T Publishing.
Lorsque plusieurs "hooks" d'un même type ont été installés, une chaîne se forme (cfr. figure 8).

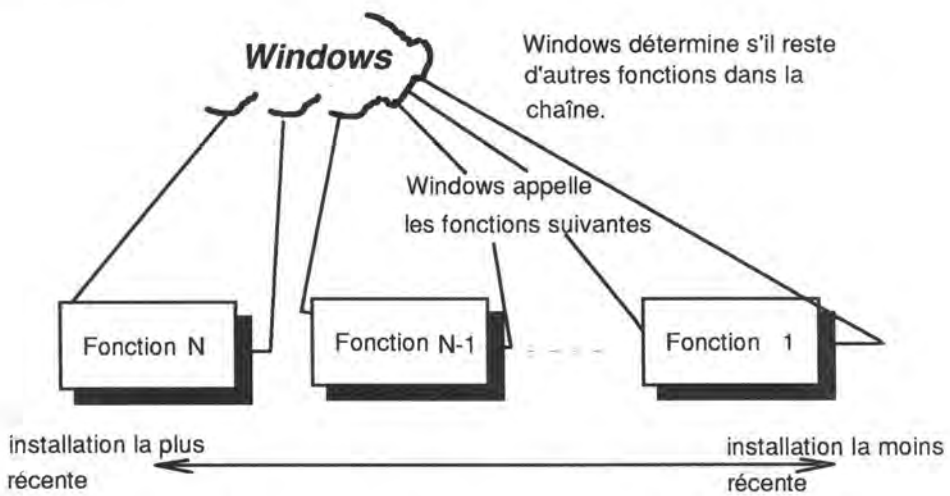


Figure 8: Le traitement des fonctions "hooks".

Source:*Windows 3.1: A Developer's Guide, Richter,1992,M&T Publishing,p371.*

Windows appelle les différentes fonctions formant la chaîne en commençant par celle qui a été installée en dernier lieu.

Il faut toutefois indiquer qu'en dépis de ces puissantes capacités, cet outil comporte quelques défauts. Le manque de claireté de la documentation, la difficulté de l'implémentation et la baisse des performances de Windows en sont quelques exemples.

Dans le logiciel auteur, deux fonctions "hooks" ont été implémentées. Elles permettent de détecter les événements relatifs aux manipulations de la souris et du clavier.

Pour chaque événement, des données telles que le temps, le type (souris, clavier, etc...), le nom de l'objet interactif sélectionné, ses coordonnées, etc. font l'objet d'un enregistrement dans un fichier.

Ce fichier pourra, par la suite, servir de base à une analyse.

3.4.Les variables

Un mécanisme permettant la manipulation de variables devenait indispensable dès que l'outil ***boîte d'édition*** (cfr chapitre 2) a été introduit dans le logiciel auteur. En effet, grâce à cet outil, l'utilisateur peut entrer une valeur (numérique ou alphanumérique) au clavier. Il fallait donc un processus sachant stocker et manipuler ces valeurs. Par ailleurs, l'utilisation de variables dans les traitements déclenchés par la sélection d'objets interactifs s'avérait être intéressante.

Du point de vue de l'implémentation, une variable est représentée par la structure de donnée suivante:

<Nom>	<Type>	<Valeur initiale>	<Valeur courante>

Figure 9: La structure de donnée représentant une variable.

- Le <nom>, composé de caractères alphanumériques, sert d'identifiant. Il est donc nécessaire de contrôler son unicité lors de la création.

- Le <type>, codé sur un byte, indique si la valeur de la variable est numérique ou alphanumérique. Il facilite le contrôle de validité des opérations effectuées sur la variable.

- La <valeur> est, quel que soit son type, représentée par des caractères alphanumériques: des méthodes de conversion sont disponibles dans le cas de variables numériques. Lors de la création de la variable, le concepteur doit indiquer sa valeur initiale. Celle-ci sera constamment gardée en mémoire pour des raisons techniques (cfr. le mécanisme de sauvegarde). Les changements de la valeur de la variable seront notées dans le champ <Valeur courante> de la structure de donnée.

Notons encore que toutes les variables sont globales et donc accessibles par tous les objets créés dans les exercices.

3.5.Le mécanisme de sauvegarde et de chargement des exercices

Le logiciel auteur n'a vraiment d'intérêt que si l'utilisateur peut enregistrer sur disque et rappeler en mémoire les exercices qu'il a conçus.

Il fallait donc trouver un mécanisme de description évitant toute perte d'information.

Comme le montre la figure suivante, un exercice est composé:

- de la table des variables,
- de la liste des descriptions de chaque écran, (la description d'un écran comprend la description des actions associées à l'écran ainsi que la description des objets interactifs composant l'écran),
- des paramètres des moyens d'interaction. (le type: clavier, souris, etc.; vitesse de déplacement, ...).

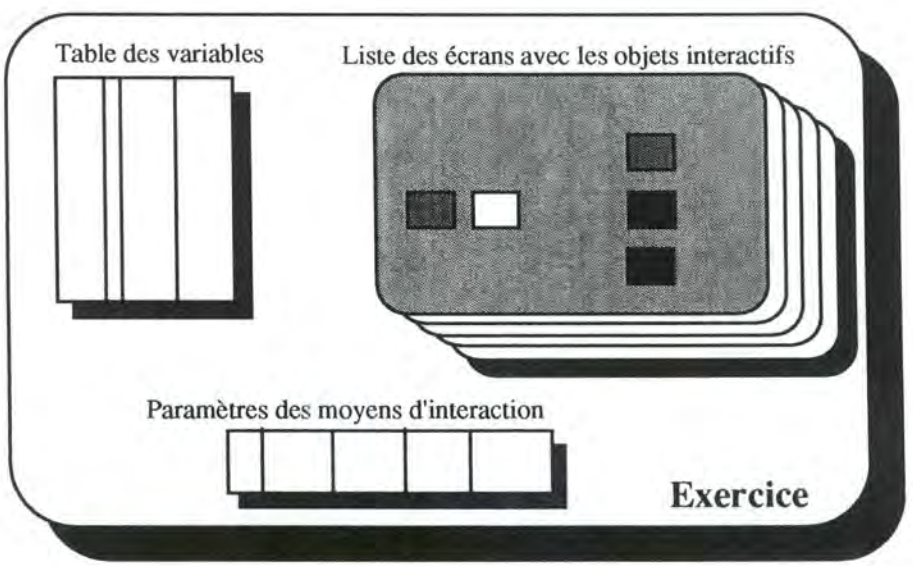


Figure 10: Les éléments constitutifs d'un exercice.

Le fichier descriptif a une partie de taille fixe et une partie de taille variable:

- La partie fixe est composée d'un en-tête donnant quelques renseignements tels que le type de fichier (confirmant qu'il a bien été créé par le logiciel auteur), la taille de la table des variables, la taille de la table des paramètres des moyens d'interaction, etc.

- La partie variable reprend la table des variables, la table des paramètres des moyens d'interaction et la description des objets composant l'exercice.

Plusieurs possibilités s'offraient quant à l'organisation de cette partie.

Dans le système, une table de correspondance fait le lien entre l'identifiant de l'objet donné par l'utilisateur et le pointeur interne à Windows. La présence de cette table facilite par exemple le contrôle de validité des identifiants lors de la création d'un nouvel objet.

Un parcours séquentiel de cette table aurait donc suffit pour enregistrer tous les objets. Toutefois, cette méthode aurait présenté un inconvénient par rapport à celle qui a été choisie.

L'idée était que l'on puisse enregistrer non seulement un exercice entier mais aussi une partie, à savoir un écran.

Pour cela, une vue orientée objet s'impose. Si chaque type d'objet est doté d'une méthode capable d'enregistrer dans un fichier les informations le concernant, le problème est réglé. Ainsi, un objet interactif enregistrera ses paramètres spécifiques et la liste des actions qui lui est associée; un écran en fera de même et "demandera" à tous les objets le composant de s'enregistrer également. Enfin, l'enregistrement d'un exercice complet se résumera à sauvegarder les tables des variables et de paramétrage et à demander à chaque écran de s'enregistrer.

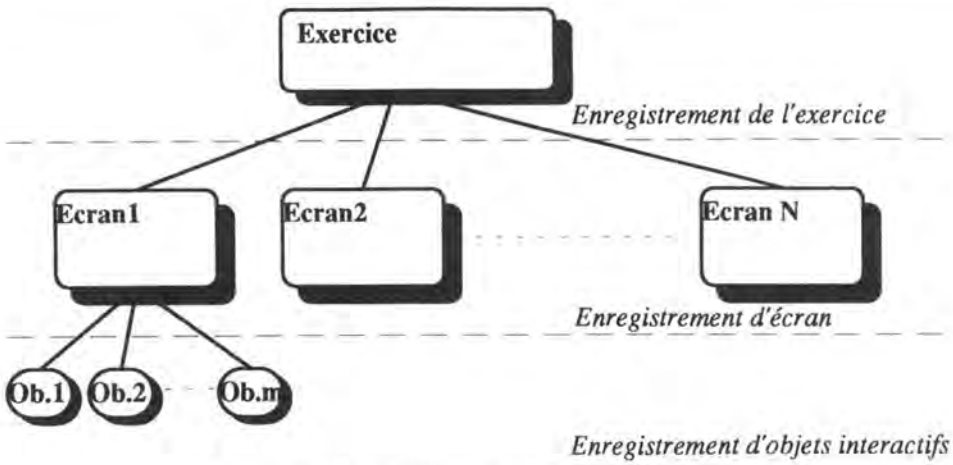


Figure 11: Les différents niveaux d'enregistrement.

La figure précédente montre les différents niveaux d'enregistrement. Chaque cadre correspond à la méthode "enregistrement" associée à l'objet. Les flèches reliant les cadres représentent un appel de la méthode "enregistrement" de l'objet pointé.

Concernant l'opération de chargement en mémoire, la lecture des tables des variables et de paramétrage ne présentent pas de difficultés majeures. C'est pourquoi je n'en parlerai pas. Attardons-nous plutôt sur la reconstruction des divers objets .

Cette reconstruction se fait en deux phases.

- La première s'occupe de la création des différents objets à partir des descriptions du fichier. La figure suivante montre les étapes suivies pour la création d'un objet à partir de sa description sur fichier.

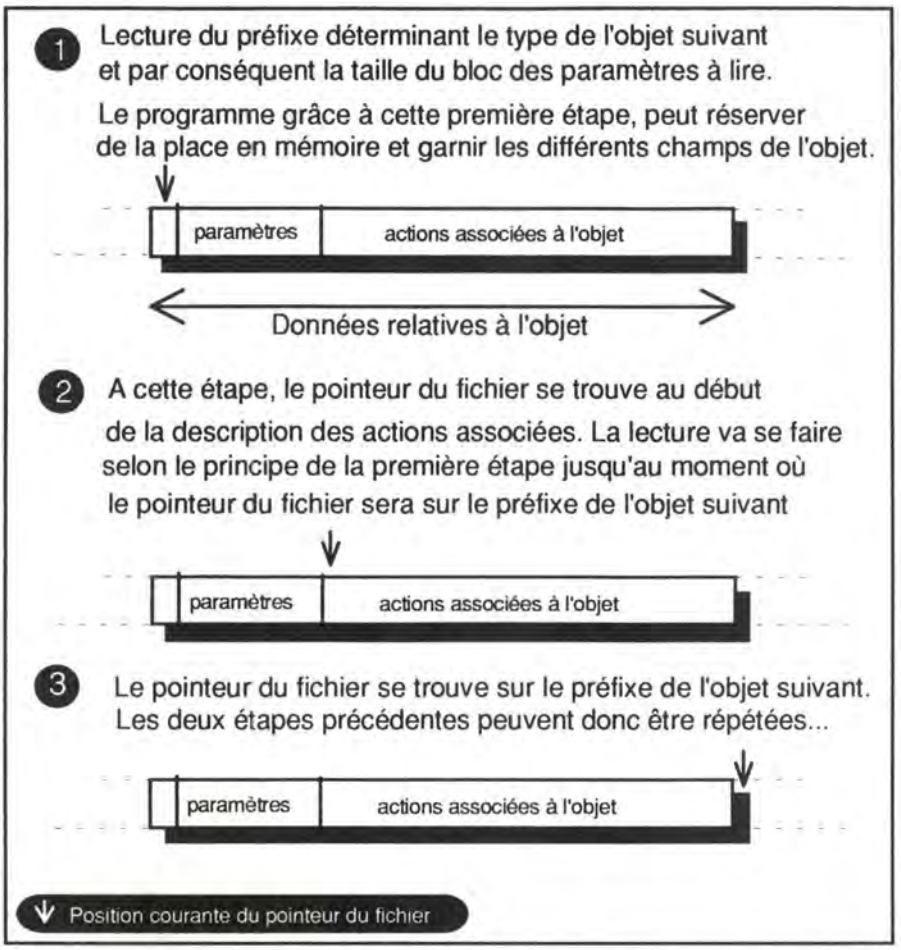


Figure 12: Les différentes étapes de la création d'un objet à partir d'un fichier.

- La deuxième phase recrée les liens éventuels entre les objets. Elle correspond à une sorte de compilation puisqu'elle consiste principalement à associer les noms des objets donnés par l'utilisateur aux pointeurs utilisés effectivement par Windows. Le fait que cela se fasse lorsque tous les objets ont été créés solutionne le problème des références en avant.

4.Conclusion

L'implémentation du logiciel auteur a permis d'aborder plusieurs points:

- ***la maintenance d'un code source*** selon ses deux aspects (les modifications et les corrections),

- ***la programmation orientée objet***,

- ***les techniques de programmation Windows*** qui demandent un certain temps d'adaptation. Une période de quatre à six mois semble nécessaire pour commencer à maîtriser les embûches de la programmation Windows.

Chapitre 7

Les étapes de construction d'un exercice utilisant ***Auteur!***

L'objectif de ce chapitre est de montrer les possibilités du logiciel ***Auteur!*** *en décrivant les étapes de la création d'un exercice type provenant du cahier des charges du Trèfle.*

1.Travail préliminaire

Le cycle de développement d'une application informatique commence souvent par l'élaboration d'un cahier des charges. Le point 6 du chapitre 3 définit, par exemple, les caractéristiques principales du logiciel ***Auteur!*** .

Les exercices qui seront développés avec ***Auteur!*** devront également être accompagnés d'un cahier des charges. Bien que les applications destinées à des personnes handicapées demandent un cahier des charges contenant une spécification très rigoureuse des fonctionnalités du logiciel ainsi qu'une description du dialogue (cfr [FRAIT,90]), rares sont ceux qui sont rédigés de manière complète dès la première élaboration. C'est souvent lors de tests d'utilisation que l'on se rend compte des nouveaux éléments modifiant ou complétant la description.

Dans notre cas, une description des différents écrans composant l'exercice semble être un minimum pour faciliter le travail "d'implémentation",

La figure suivante montre l'exercice qui va servir d'exemple tout au long du chapitre.

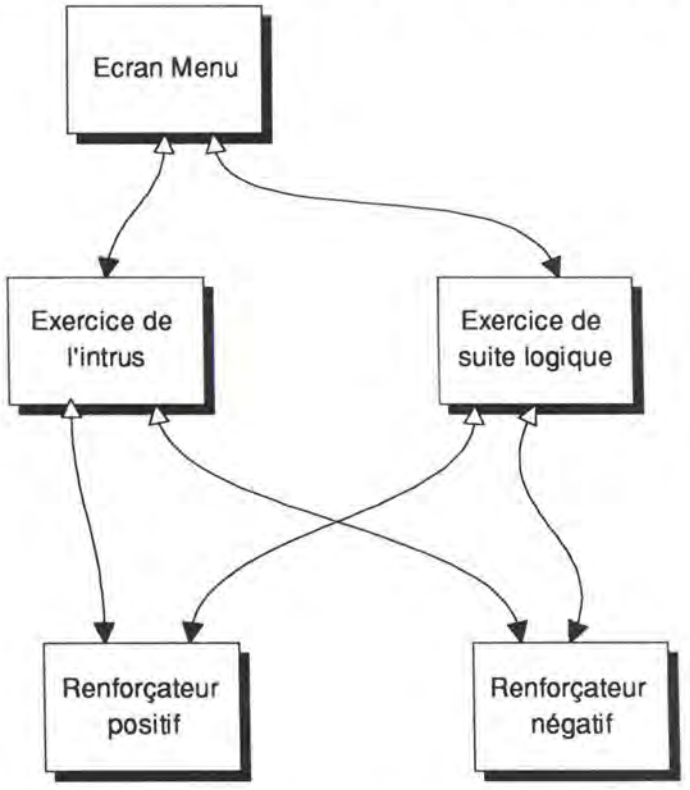


Figure 1: Les interactions entre les différents écrans de l'exercice choisi.

Dans cette figure, chaque rectangle représente un écran. Les flèches reliant les rectangles symbolisent un appel d'écran. Les flèches noires pointent vers les écrans "appelés" alors que les blanches indiquent que l'on peut revenir vers l'écran "appelant".

L'exercice que nous allons élaborer comporte donc 5 écrans:

- un écran présentant l'exercice de recherche d'intrus (cfr. chapitre 1),
- un écran présentant un exercice de suite logique (cfr. la description ci-dessous),
- un écran servant de menu,
- un écran proposant un renforçateur positif,
- un écran proposant un renforçateur négatif.

Dans cette "mini-application", l'utilisateur pourra donc choisir parmi deux exercices. Lorsqu'il répondra correctement, il déclenchera le renforçateur positif. Dans le cas contraire, ce sera le renforçateur négatif qui sera déclenché.

Un écran de renforçateur sera composé d'une petite animation et d'un bouton permettant le retour vers l'écran appelant. Une musique, gaie ou triste selon le type de renforçateur, animera ces deux écrans.

Voyons maintenant la description de l'exercice de suite logique:

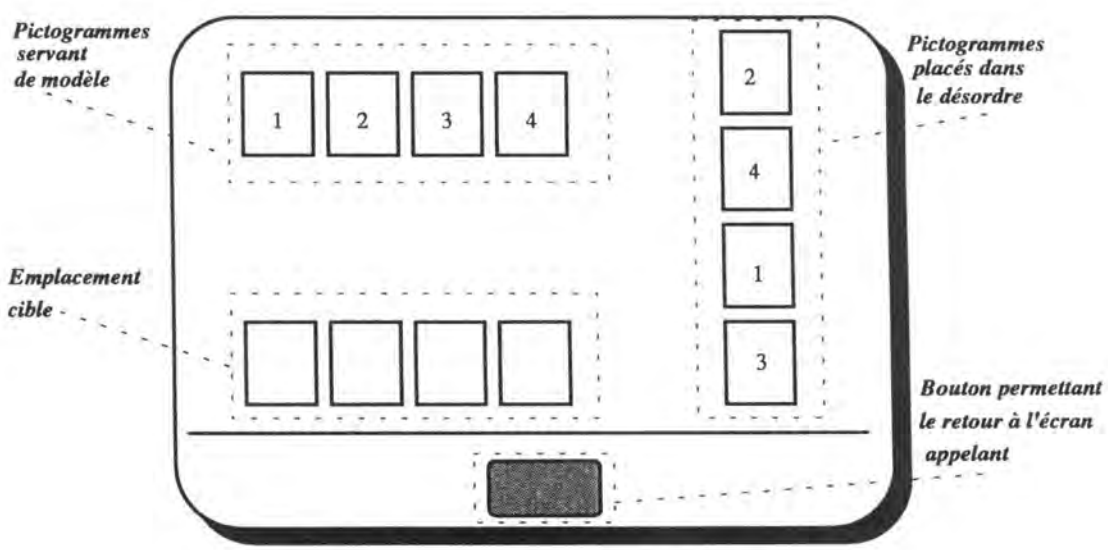


Figure 1: L'interface de l'exercice de suite logique.

Le principe de cet exercice est de remettre dans le bon ordre la séquence de pictogrammes présentés à droite de l'écran. Pour aider l'enfant, la séquence correcte est affichée en haut de l'écran. Lorsque l'enfant sélectionne le bon pictogramme, celui-ci se déplace jusqu'à l'emplacement cible qui lui est réservé. Si le pictogramme sélectionné

n'est pas le bon, le renforçateur négatif est appelé. Lorsque la séquence est complètement reconstituée, c'est le renforçateur positif qui est appelé.

L'écran servant de menu est composé quant à lui de deux boutons dont la sélection appelle soit l'exercice de recherche d'intrus, soit l'exercice de suite logique.

Avant de montrer comment "implémenter" cet exercice avec ***Auteur!***, relevons les différents éléments dont nous aurons besoin.

- Pour l'exercice de l'intrus, quatre pictogrammes auquel on ajoute un pictogramme représentant le "retour à l'écran du menu".
- Pour l'exercice de la suite logique, nous avons également besoin de quatre pictogrammes en plus du pictogramme "retour à l'écran du menu".
- Pour l'écran du menu, deux pictogrammes représenteront les exercices disponibles.
- Enfin, pour chaque renforçateur, nous aurons besoin de plusieurs pictogrammes composant l'animation. Nous aurons également besoin de fichiers sonores.

Nous supposerons, dans la suite du chapitre, que nous disposons de ces différents éléments[1].

2.L'implémentation de l'exercice

2.1.Présentation de l'écran de Auteur!

La figure suivante montre l'interface du logiciel ***Auteur!*** .

[1]*Nous pouvons, par exemple, les obtenir à partir de librairies de pictogrammes fournies avec d'autres logiciel de dessins.*

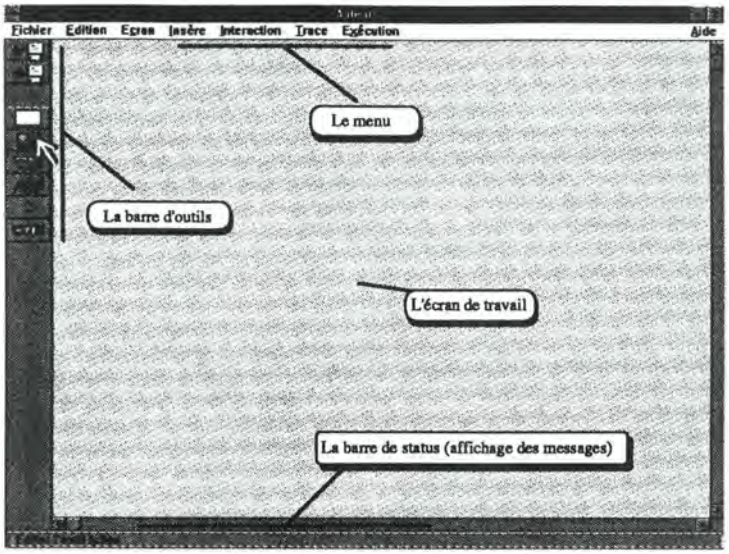
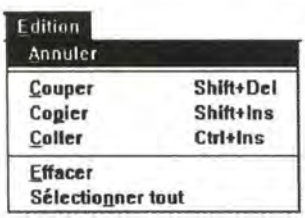


Figure 3:L'écran de ***Auteur!***.

Auteur! est composé de quatre parties principales:

- l'***écran de travail*** sur lequel le concepteur pourra placer les objets interactifs composant les exercices à développer,
- la ***barre d'outils*** reprenant les différents outils cités au chapitre 3,
- la ***barre de menu*** donnant accès aux divers sous-menus:

permet:
- de créer un nouvel exercice,
- d'ouvrir un fichier existant,
- d'enregistrer les exercices
- de définir certaines options.



permet:
- d'annuler la dernière modification apportée,
- de couper, copier et coller des objets. Il s'agit des opérations liées au presse-papier de Windows.

permet:
- de se déplacer parmi les écrans définis dans l'exercice développé.

permet:
- d'insérer les différents objets définis dans la boîte à outils.

Variables — permet:
- d'appeler la boîte de dialogue gérant la création et la destruction des variables.

Interaction — permet:
- d'appeler la boîte de dialogue gérant les interactions.

Trace — permet:
- de gérer la trace.

Exécution — permet:
- de basculer dans le mode exécution.

Aide / Index / A propos de... — permet:
- d'obtenir de l'aide (non implémentée),

- la ***barre d'état*** affichant les messages destinés au concepteur.

2.2.La création des écrans

Procédons maintenant à la création des écrans. Seuls, celui du renforçateur positif et celui de l'exercice de suite logique seront développés ici afin d'éviter toute répétition.

2.2.1.Le renforçateur positif

2.2.1.1.placement des objets

La méthode de placement d'objet a été décrite dans le point 5.4.3.1. du chapitre 3.
Pour ce renforçateur, nous devons utiliser deux objets: l'animation et l'icône permettant le retour vers l'écran "appelant". L'insertion de ces deux objets se fait

soit par le choix des items *Insère-Animation* et *Insère-Icône* du menu, soit de manière directe en cliquant sur les icônes correspondantes de la barre à outils. Pour déplacer un objet, il suffit de le sélectionner (en cliquant dessus) et de faire glisser la souris tout en maintenant le bouton gauche enfoncé.

2.2.1.2.paramétrage

Le paramétrage se fait en cliquant deux fois très rapidement sur l'objet. Une boîte de dialogue apparaît alors. La figure suivante montre la boîte associée à l'animation. Nous pouvons y définir le nom, la position et les dimensions de l'objet, déterminer s'il peut être sélectionné ou non, et régler la vitesse de défilement des images de l'animation.
Deux boutons supplémentaires permettent la sélection des images composant l'animation. Celui nommé "*Ajoute*" fait apparaître la boîte de dialogue de sélection de fichiers illustrée à la figure 5 alors que celui libellé "Supprime" retire l'élément courant de la liste des fichiers.

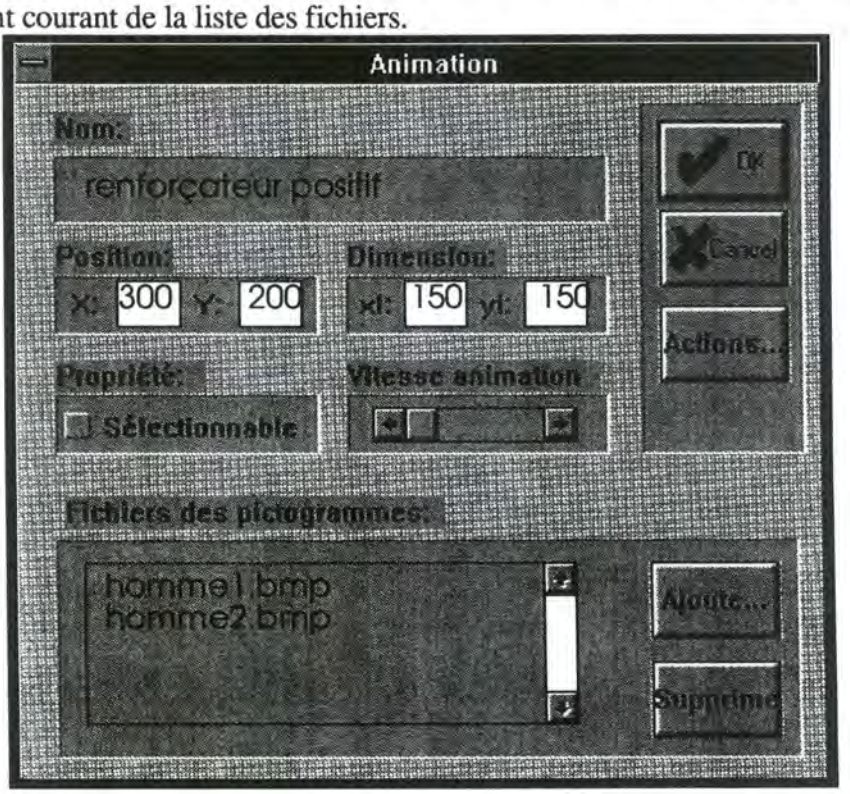


Figure 4:Le paramétrage de l'animation.

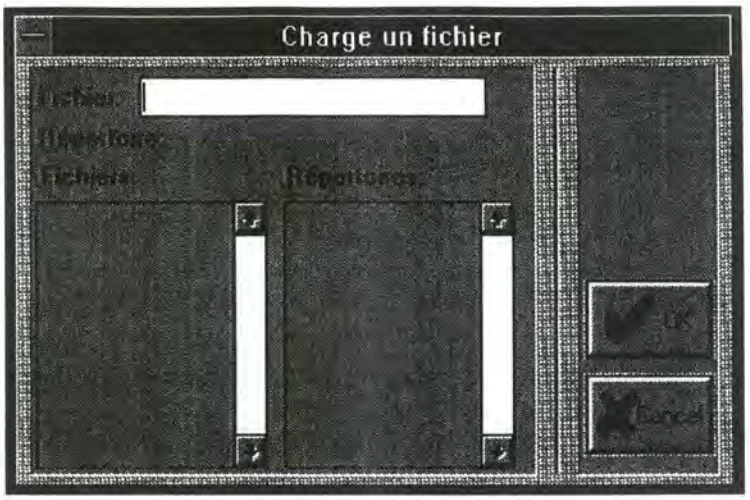


Figure 5:La boîte de sélection de fichiers.

La boîte de dialogue attachée à l'icône a déjà été présentée à la figure 21 du chapitre 3. L'icône permettra le retour à l'écran "appelant". Il faudra donc cocher la propriété "*sélectionnable*" et associer à l'icône l'action "*appel d'écran*". Les deux figures suivantes montrent la procédure à suivre:
La première figure représente la boîte de dialogue qui apparaît lorsque l'on appuie sur le bouton "*Actions*"[2].

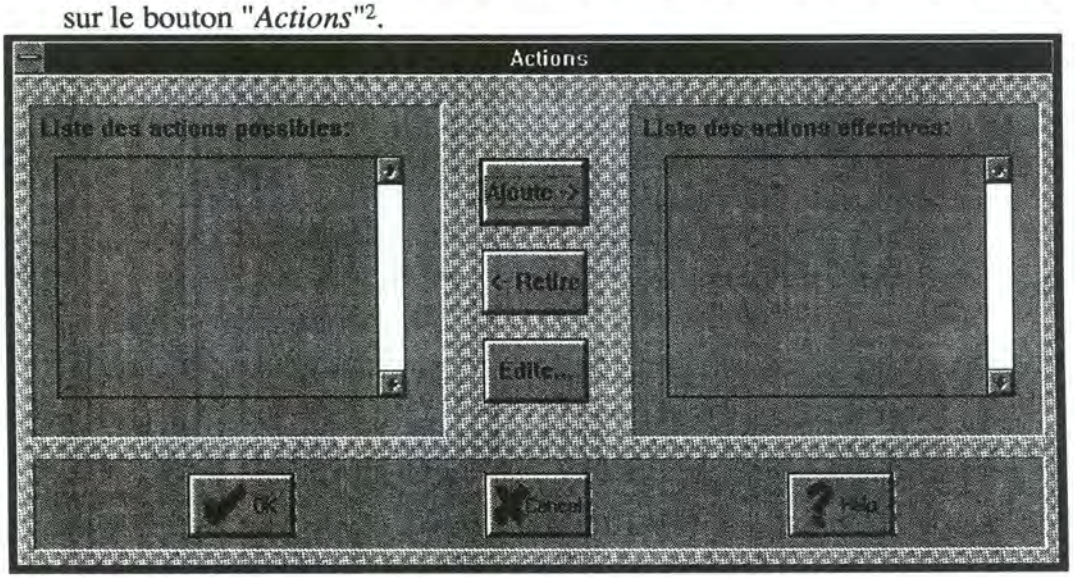


Figure 6:La définition de la séquence d'action associée à un objet.

Les deux boutons "*Ajoute*" et "*Retire*" permettent, comme leur nom l'indique, d'ajouter et de retirer une action de la liste effective.
Le bouton "*Edite*" affiche, quant à lui, la boîte de dialogue associée à l'action courante de la liste effective.

---

[2]*Ce bouton est présent dans chaque boîte de dialogue permettant le paramétrage des objets de la boîte à outils (cfr la figure 21 du chapitre 3 ou les figures 4 et 8 de ce chapitre).*

Dans le cas qui nous intéresse, nous allons ajouter l'action "*Appel d'écran*" et l'éditer. Il suffira alors de cocher le bouton radio "*Retour vers l'écran précédent*" et de refermer la boîte de dialogue.

Figure 7:La définition de l'action "*retour à l'écran précédent*".

Les éléments constituant l'écran du renforçateur sont maintenant définis. Il reste encore à déterminer quelques caractéristiques propres à l'écran: sa couleur et la séquence d'actions à exécuter lorsqu'il apparaît.

Pour ce faire, un double clic sur l'écran présentera la boîte de dialogue de la figure 8.

Figure 8:Le paramétrage de l'écran.

La sélection du bouton "*Couleur*" fera apparaître la boîte de dialogue suivante qui permettra en deux clics sur la couleur de votre choix et sur le bouton "*Ok*" de repeindre le fond de l'écran.

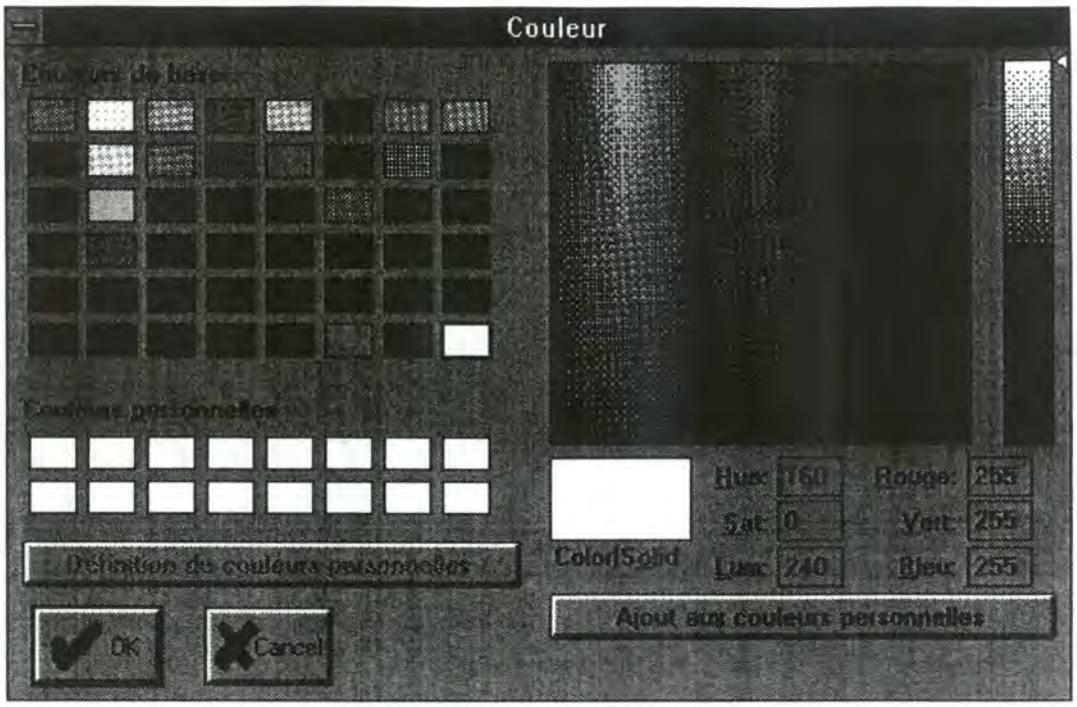


Figure 9:Boîte de dialogue donnant un large choix de couleurs.

Le bouton "*Actions*" ouvrira, quant à lui, la boîte de dialogue présentée à la figure 6.

Les actions à associer à cet écran sont d'une part le déclenchement de l'animation et d'autre part l'accompagnement d'une petite mélodie.

Les deux boîtes de dialogue suivante montre comment déterminer ces actions:

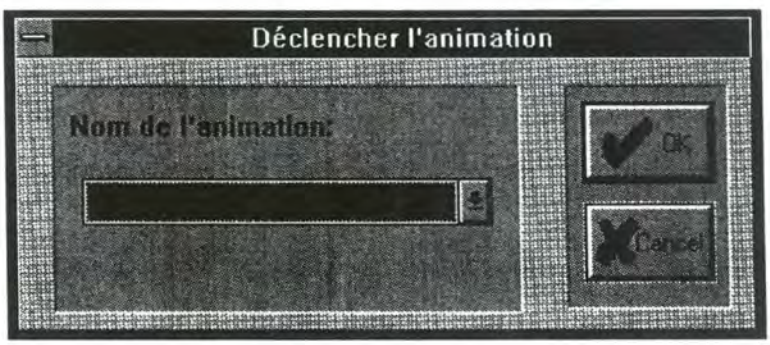


Figure 10: Le déclenchement de l'animation.

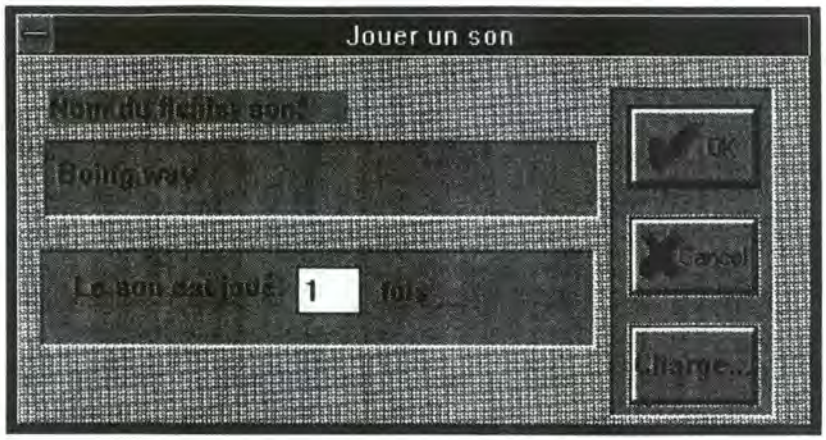


Figure 11: La sélection d'une mélodie.

L'autre renforçateur se crée selon le même principe.

2.2.2.L'exercice de suite logique

Le placement et le paramétrage des éléments ayant été expliqués dans le point précédent, nous allons plutôt centrer celui-ci sur le concept de variables.

Dans cet exercice, nous avons besoin de savoir à tout moment quelle est l'icône qui doit être sélectionnée (cfr. description de l'exercice en début de chapitre). Si l'on clique sur une icône, celle-ci doit être déplacée uniquement dans le cas où elle correspond à l'ordre de la suite logique.

Nous pouvons réaliser cela en créant une variable que l'on nommera par exemple "icône courante".

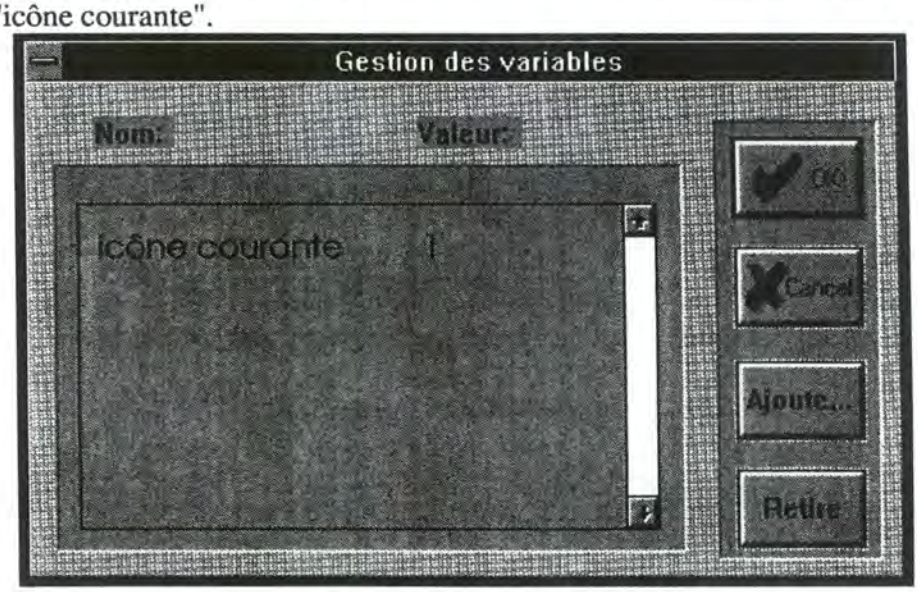


Figure 12: La création de la variable.

L'action associée à chaque icône consistera alors à comparer le "numéro d'ordre" de cette icône à la valeur de la variable . S'il y a égalité, l'icône devra être déplacée vers sa nouvelle position et il faudra incrémenter la valeur de la variable. Les figures suivantes indiquent comment réaliser ces actions:

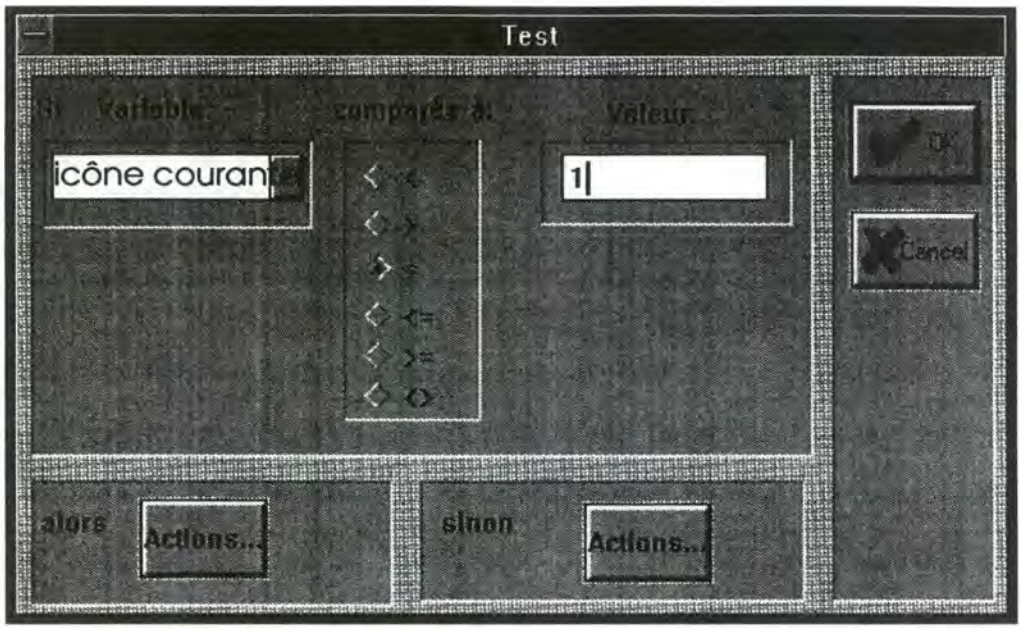


Figure 13: Test d'égalité entre le numéro d'ordre et la valeur de la variable.

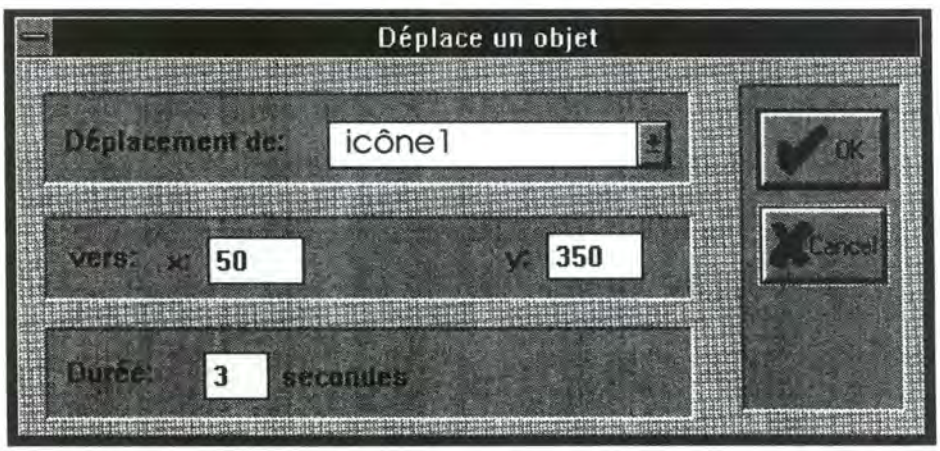


Figure 14: Le déplacement d'un objet vers une nouvelle position.

Figure 15: L'incrémentation de la variable.

2.3.L'établissement des liens entre les écrans

Lorsque les différents écrans de la "mini-application" ont été créés, il faut encore les relier entre eux.

Cette liaison s'effectue grâce à l'action "appel d'écran" qui a déjà été présentée à la figure 7.

Si on se réfère à la figure 1, nous pouvons déterminer ces liens.

- Chaque élément de l'écran "*menu*" appelle un exercice.
- L'élément "*intrus*" de l'exercice de recherche d'intrus appelle le "*renforçateur positif*" alors que les autres éléments appellent le "*renforçateur négatif*".
- Le dernier élément de la suite logique appelle le "*renforçateur positif*" tandis que les éléments sélectionnés au mauvais moment appellent le "*renforçateur négatif*".

Rappelons encore que les "*retours à l'écran appelant*" ont déjà été définis lors de la création de chaque écran.

2.4.Les moyens d'interaction

Avant de passer dans le mode exécution, il convient de choisir les moyens d'interaction qui conviendront au mieux aux enfants qui utiliseront la "mini-application". Nous avons déjà montré aux figures 14 et 15 du chapitre 3 comment se faisait le paramétrage de la souris, voyons maintenant comment utiliser le défilement automatique.

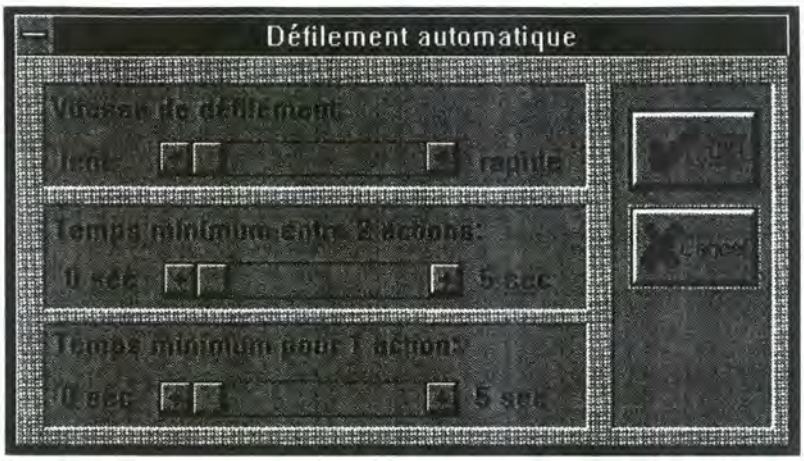


Figure 16: Les paramètres du défilement automatique.

Les paramètres du défilement sont:

- La vitesse du défilement.
- Le temps minimum entre deux actions pour que la seconde action soit prise en compte.
- La durée minimum d'une action pour qu'elle soit prise en compte.

Notons encore que le type de défilement implémenté correspond au mécanisme utilisant un seul interrupteur. Dans ***Auteur!***, cet interrupteur est simulé par la barre d'espace.

2.5.La trace

Nous pouvons également choisir, avant de passer dans le mode exécution, d'enclencher ou non le mécanisme de trace.

La figure 16 du chapitre 3 présente la boîte de dialogue en question.

2.6.L'exécution

Nous pouvons maintenant basculer en mode *exécution* (cfr. menu) et laisser la place à l'enfant.

Tous les éléments permettant l'élaboration de l'exercice disparaissent de l'écran. Si le mécanisme de trace a été déclenché, chaque événement du système va s'enregistrer dans le fichier spécifié. Si le mécanisme de défilement a été sélectionné, le curseur commencera à défiler dès ce moment.

L'éducateur qui encadre l'enfant peut à tout moment stopper l'exercice et revenir en mode conception afin de porter des modifications ou tout simplement changer d'exercice.

3.Conclusion

Nous n'avons voulu reprendre dans ce chapitre toutes les possibilités de ***Auteur!*** L'objectif était plutôt de montrer la procédure générale à suivre pour la construction d'un exercice, pas à pas.

Il faut souligner l'importance du travail préliminaire qui consiste en l'élaboration du cahier des charges. Celui-ci facilite par la suite l'"implémentation" de l'exercice.

Le temps consacré à l'implémentation est quant à lui réduit fortement puisque, si l'on dispose des différents pictogrammes et sons à inclure dans l'exercice, celui-ci est implémenté en moins d'une heure.

Chapitre 8

Perspectives

La première version d'un logiciel a souvent besoin d'être remaniée. Ce chapitre propose des modifications ou améliorations envisageables qui n'ont pu être implémentées par manque de temps.

Le contrôle de validité

Auteur! est un outil fort général qui laisse par conséquent une grande marge de liberté au concepteur des exercices. Toutefois, cette liberté doit être contrôlée afin d'éviter des incohérences dans la description des exercices.

Prenons par exemple une icône sélectionnable dont l'action associée est l'appel d'un écran de l'exercice (cfr. chapitre 7). Si, entre le moment où nous avons décrit cette action et le passage en mode *exécution* , nous effaçons l'écran qui doit être appelé, la description du système devient incohérente.

Il convient donc de vérifier la cohérence du système décrit avant de passer en mode *exécution* même si cette vérification coûte en temps.

L'amélioration de l'interface

Des progrès peuvent être réalisés. Prenons par exemple le cas de la gestion des actions associées aux objets. Cette gestion se fait, dans l'état actuel des choses, par la boîte de dialogue décrite à la figure 5 du chapitre 7. Des listes de sélection sont utilisées pour représenter la séquence d'actions effectives.

Une façon, plus visuelle, de représenter cette séquence imiterait le mécanisme de description d'algorithmes tel qu'utilise ***Authorware Professional*** (cfr chapitre 3).

Des icônes, représentant chacune une action, pourraient être placées sur une ligne décrivant la séquence à suivre (cfr figure 1). En cliquant sur une icône, on obtiendrait la boîte de dialogue permettant le paramétrage de l'action en question.

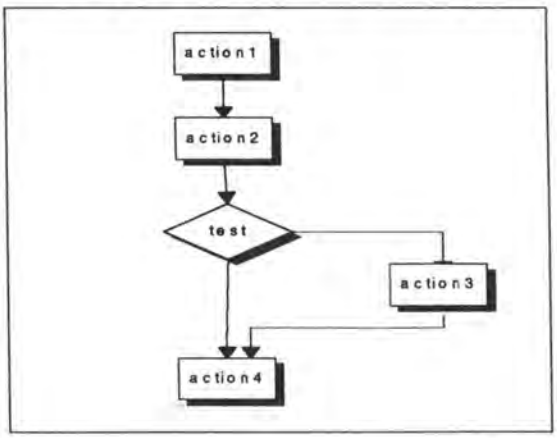


Figure 1:Description des actions à la manière d'un algorithme.

L'ajout d'outils et d'actions

L'utilisation du logiciel ***Auteur!*** par les éducateurs révèlera si la boîte à outils proposée est complète ou non.

On pourrait par exemple ajouter des outils permettant la création de dessins (rectangles, lignes, etc.).

Des séquences vidéo pourraient, moyennant l'acquisition d'une carte vidéo, être gérées par ***Auteur!***.

La gestion de la trace

La gestion de la trace se limite dans la version actuelle à l'enregistrement des événements déclenchés durant les sessions en mode *exécution*.

Une amélioration à envisager pourrait être l'introduction de mécanismes d'analyse de ces enregistrements.

La génération de rapport sur imprimante

Il serait intéressant de fournir la description des exercices implémentés dans Auteur! sous la forme d'un rapport reprenant les différents écrans ainsi que leurs objets, les variables utilisées etc.

Ce rapport serait d'un grande aide, sans aucun doute, lorsque des modifications devraient être apportées aux exercices.

L'aide

Un système d'aide, reprenant au minimum un index des outils et des actions du logiciel pourrait être implémenté. On pourrait également ajouter dans ce système d'aide des exemples d'utilisation des différentes parties du logiciel.

Conclusion

Par ce chapitre, nous avons essayé de montrer que le projet, bien qu'il soit déjà bien avancé , était loin d'être terminé et qu'on pouvait y apporter de nombreuses extensions.

Conclusion

Le projet qui a conduit à l'élaboration du logiciel ***Auteur!*** a permis d'aborder des domaines fort diversifiés.

Etant donné la spécificité de la population concernée par le logiciel, nous avons procédé, durant le stage, à une évaluation des différents outils développés l'an passé. Bien que l'on ne puisse pas considérer cette évaluation comme complète - il aurait fallu pour cela l'entreprendre d'une part sur une population plus nombreuse et d'autre part durant une période plus grande - celle-ci a conduit à plusieurs modifications des outils.

Afin de rendre ces outils directement utilisables par des personnes non spécialisées en informatique, nous avons dû aborder le domaine des logiciels auteurs. Nous avons fait un tour d'horizon de ce qui existait sur le marché. Aucun produit commercialisé ne remplissait les conditions adéquates pour être retenu. Nous avons décidé de concevoir et d'implémenter un nouveau logiciel: ***Auteur!***.

Pour ce faire, nous avons utilisé une méthode de conception orientée objet: OBLOG. L'emploi de cette méthode assurait d'une part une certaine homogénéïté par rapport à la description des outils développés l'an passé. D'autre part, elle permettait d'obtenir certaines qualités de conception que la littérature attribue fréquemment aux méthodes orientées objet.

Une de ces qualités est ce qu'on appelle la réutilisabilité. Celle-ci exige une certaine généralité des modules développés et est certainement facilitée par les mécanismes d'héritage offerts par les méthodes orientées objet.

Les modules du logiciel ***Auteur!*** présentent ce caractère de réutilisabilité. Nous les avons d'ailleurs utilisés dans un autre projet développé dans le cadre du cours de simulation de systèmes donné en troisième licence par Madame Noirhomme.

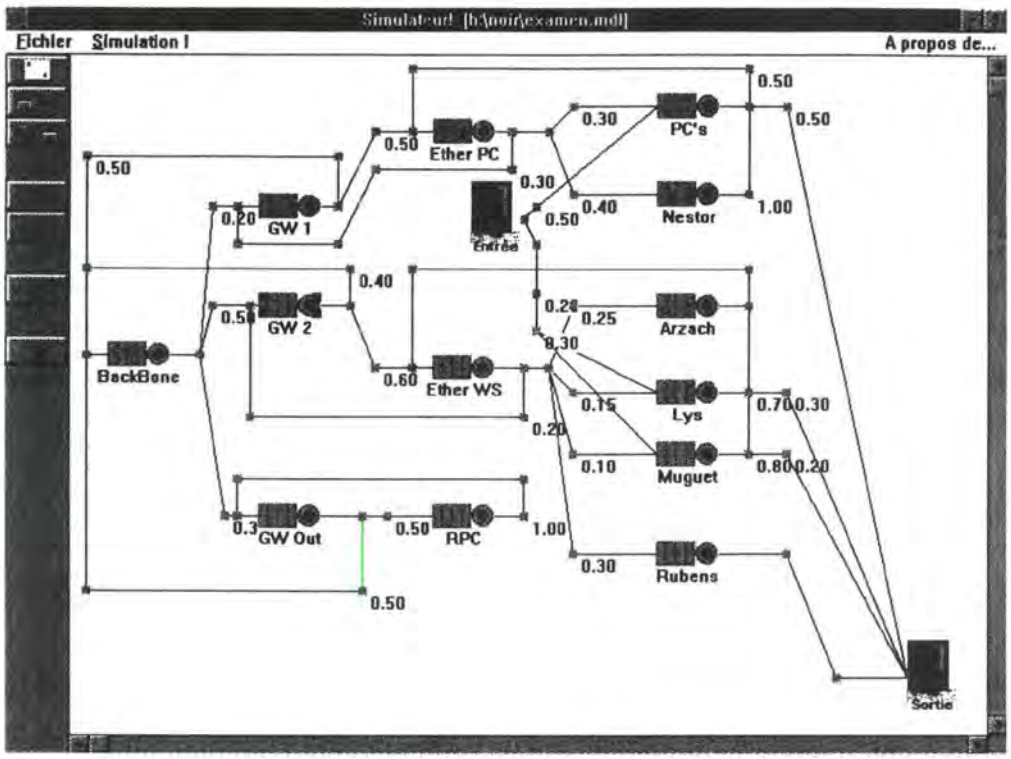


Figure 1: L'interface du logiciel de simulation.

Source: [BOUCH,93], page 18.

Ce projet présentait en effet plusieurs caractéristiques communes à ***Auteur!***.
Dans cette application, la construction d'un modèle de simulation de réseau s'effectue en plaçant sur l'écran les différents éléments constitutifs dont le paramétrage est défini par des boîtes de dialogue propres à chaque élément. L'utilisateur peut ainsi définir la politique de la file d'attente de chaque serveur, sa longueur et la loi statistique gérant le serveur. Les probabilités de transition entre les différents serveurs sont également décrites de manière très visuelle. La figure 1 montre un exemple de réseau modélisé avec le simulateur. Les résultats fournis par le logiciel indiquent des intervalles de confiances de temps d'attente moyen, de nombre moyen de clients par serveur, etc. Le lecteur intéressé pourra trouver de plus amples renseignements sur ce projet dans [BOUCH,93].

Nous terminerons en rappelant une fois de plus que l'application ***Auteur!*** a été développée sous l'environnement ***Windows*** et que celui-ci nécessite une longue

période d'apprentissage pour tous ceux qui désirent maîtriser la programmation sous cet environnement.

Plusieurs extensions possibles du projet viennent à l'esprit, elles ont d'ailleurs fait l'objet d'un chapitre, mais nous pensons que cette première version de ***Auteur!*** a jeté les bases originales d'un logiciel auteur destiné au domaine du handicap.

Bibliographie

Ouvrages:

[Brou,90] ,L. Brousmiche, X. Gillet, *Logiciel de simulation d'un jeu de marionnettes pour enfants infirmes moteurs cérébraux*, Mémoire F.U.N.D.P., 1990.

[Colv,90] ,D. Colven, *A common terminology for switch controlled software*, ACE Centre, Oxford, 1990.

[Démo,92] ,R. Démo, S. Baudrenghien, *Elaboration d'une boîte à outils d'aide à la réalisation d'interfaces pour personnes handicapées*, Mémoire F.U.N.D.P., 1992.

[Depl,89] ,M. Deplechin, G. Strappazzon, *Un logiciel de gestion des comptes pour personnes handicapées mentales adultes, développement et évaluation*, Mémoire F.U.N.D.P., 1989.

[Jenn,92] ,R. Jennings, *Discover Windows 3.1. Multimedia*, Que Corporation, Carmel, 1992.

[Lebl,92] ,G. Leblanc, *Programmation Windows en Turbo C++ et Borland C++*, Eyrolles, France,1992.

[Lep,90] ,T. Lepoutre, J.M. Roquet, *La personne handicapée mentale et la connaissance du corps humain: un logiciel d'apprentissage*, Mémoire F.U.N.D.P., 1990.

[Lien,80] ,B.P. Lientz, *Software Maintenance Management*, Addison-Wesley, England, 1980.

[Lovi,92] ,J-P. Lovinfosse, *Les best-sellers de l'informatique, Windows 3.1 et le multimédia*, Marabout, Belgique,1992.

[Meye,88] ,B. Meyer, *Object Oriented Software Construction*, Prentice-Hall International, New York, 1988.

[Rich,92] J.M. Richter, *Windows 3.1.: A developer's guide*, M&T Publishing, ,1992.

[Schu,92] ,A. Schulman et al., *Les coulisses de Windows*, Addison-Wesley, France, 1992.

[Terr,91] ,Fr. Terrier, *Borland C++ Concepts fondamentaux*, volumes 1 et 2, Sybex, France, 1991.

[Vand,91] ,J. Vanderdonckt et al., *Une description orientée objet des objets interactifs abstraits utilisés en interface Homme-Machine*, Institut d'Informatique, 1991.

[Wood,90] ,N. Woodhead, *Hypertext & Hypermedia, theory and applications*, Addison-Wesley, England, 1990.

[Zeip,92] ,J.M. Zeippen et al., *Pragmatic introduction to the OBLOG Approach in System Design*, notes de cours de Méthodologie de développement de logiciels, matières appronfondies, ESDI, Lisbon, 1992.

Articles:

[Auth,91] ,X, *Authorware professional for Windows*, brochure publicitaire, Octobre 1991.

[Bouch,93] ,O. Bouchez,L. Michel, L.Vandenabeele, *La simulation, de la théorie à l'implémentation*, travail effectué dans le cadre du cours de Simulation de systèmes donné par Madame M. Noirhomme. 3e licence, Institut d'Informatique, 1993.

[Drum,93] ,X, Drum, *Software support for video-assisted evaluation of usability*, brochure explicative, April 1993.

[Four,93] ,G. Fouchard, Logiciels Auteurs, in Multimedia Solutions, Mars 93, pp.49-58.

[Frait,90] ,M. Fraiture, C. Machgeels, *Le cahier des charges d'un logiciel pour personnes handicapées*, Hantépsycom,Kerpape, 1990.

[Kiyo,93] ,G. Kiyooka, *Des DLLs orientées objets*, in Micro Systèmes, Janvier 93, pp.124-126.

[Lazz,93] ,J. Lazzaro, *Computers for the Disabled*, in Byte June 93, pp.59-64

[Link,90] ,X, *IBM LinkWay 2.0.*, brochures publicitaire,IBM Corporation, 1990.

[Music,93] ,X, *Music Metrics for Usability Standards in Computing*, Music, Consortium, March 93.

[Noir,91] ,M.Noirhomme-Fraiture, *Evaluation of software for people with mental disabilities*, World Congress on Technology, Washington, 1991.

[Sanz,92] ,D. Sanz, et al., *Préparez-vous au multimédia*, in SVM, Mars 1992, pp.69-85.

[Stor,90] ,X, *IBM Storyboard Live*, brochures publicitaire,IBM Corporation, 1990.

[Svan,93] ,D. Svanaes, *Comspec: a software architecture for users with special needs*,University of Trondheim,Norway, 1993.

[Weil,93] ,P.Weiler, *Software for the Usability Lab: a Sampling of Current Tools*,in Interchi'93, April 1993, pp.57-60.

[Wind,93] ,X, *Windows,Windows Everywhere?*, in Byte june 1993, pp. 72-94